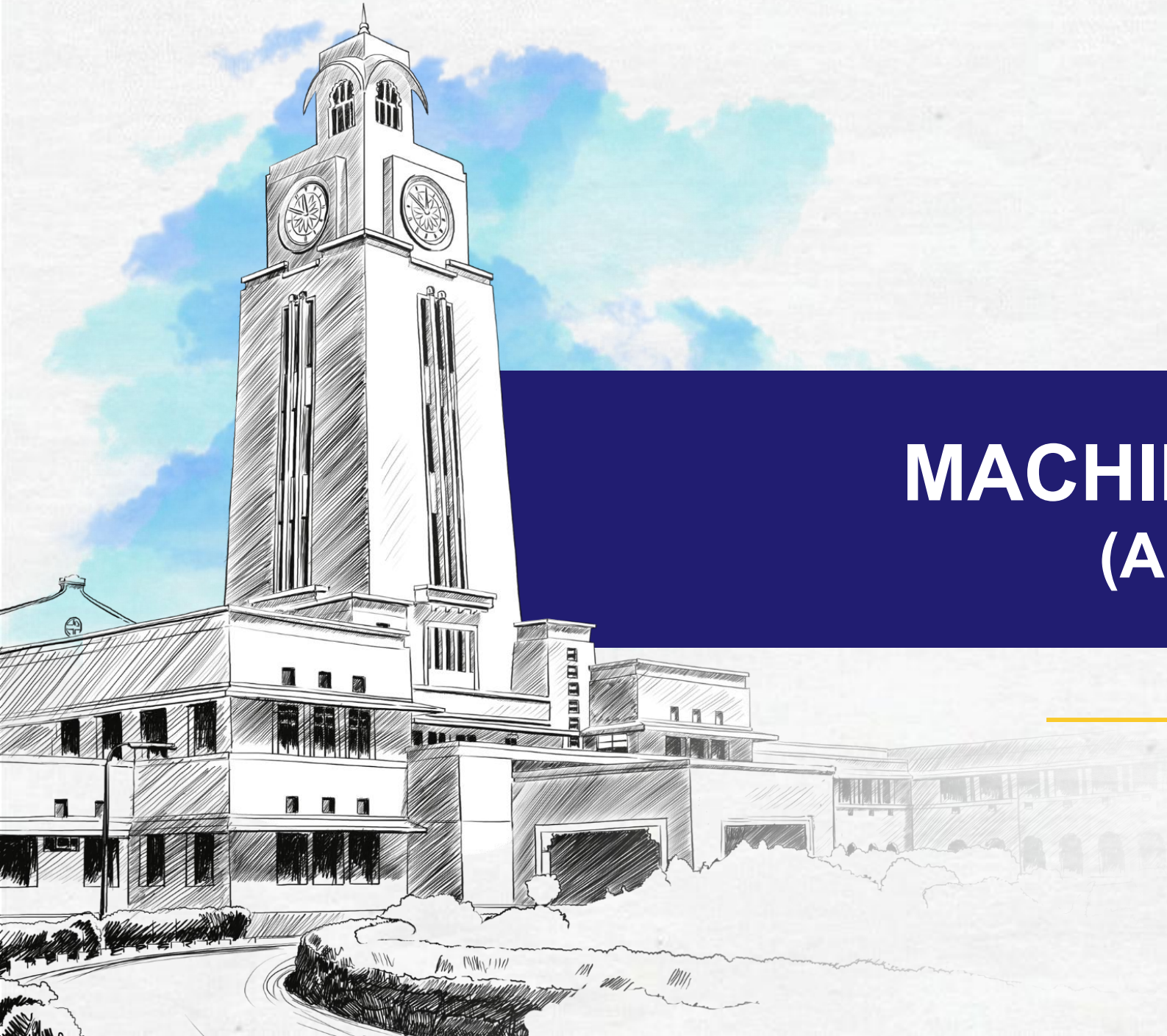


BITS Pilani
Pilani | Dubai | Goa | Hyderabad

MACHINE LEARNING
(AIML ZG565)

Session 3&4
(8th &14th Feb,2026)

Course Plan

M1	Introduction
M2	Machine learning Workflow
M3	Linear Models for Regression
M4	Linear Models for Classification
M5	Decision Tree
M6	Instance Based Learning
M7	Support Vector Machine
M8	Bayesian Learning
M9	Ensemble Learning
M10	Unsupervised Learning
M11	Machine Learning Model Evaluation/Comparison

Agenda

- Linear Model for Regression
- Direct solution vs Iterative Method
- Gradient Descent
- Linear Basis Function
- Notion of Bias vs Variance

Linear Regression

Inductive Learning Hypothesis : Interpretation

- Target Concept
- Discrete : $f(x) \in \{Yes, No, Maybe\}$ Classification
- Continuous : $f(x) \in [20\text{-}100]$ Regression
- Probability Estimation : $f(x) \in [0\text{-}1]$

Sky	AirTemp	Altitude	Wind	Water	Forecast	Humidity
Sunny	Warm	Normal	Strong	Warm	Same	60
Sunny	Warm	High	Strong	Warm	Same	75
Rainy	Cold	High	Strong	Warm	Change	70
Sunny	Warm	High	Strong	Cool	Change	45

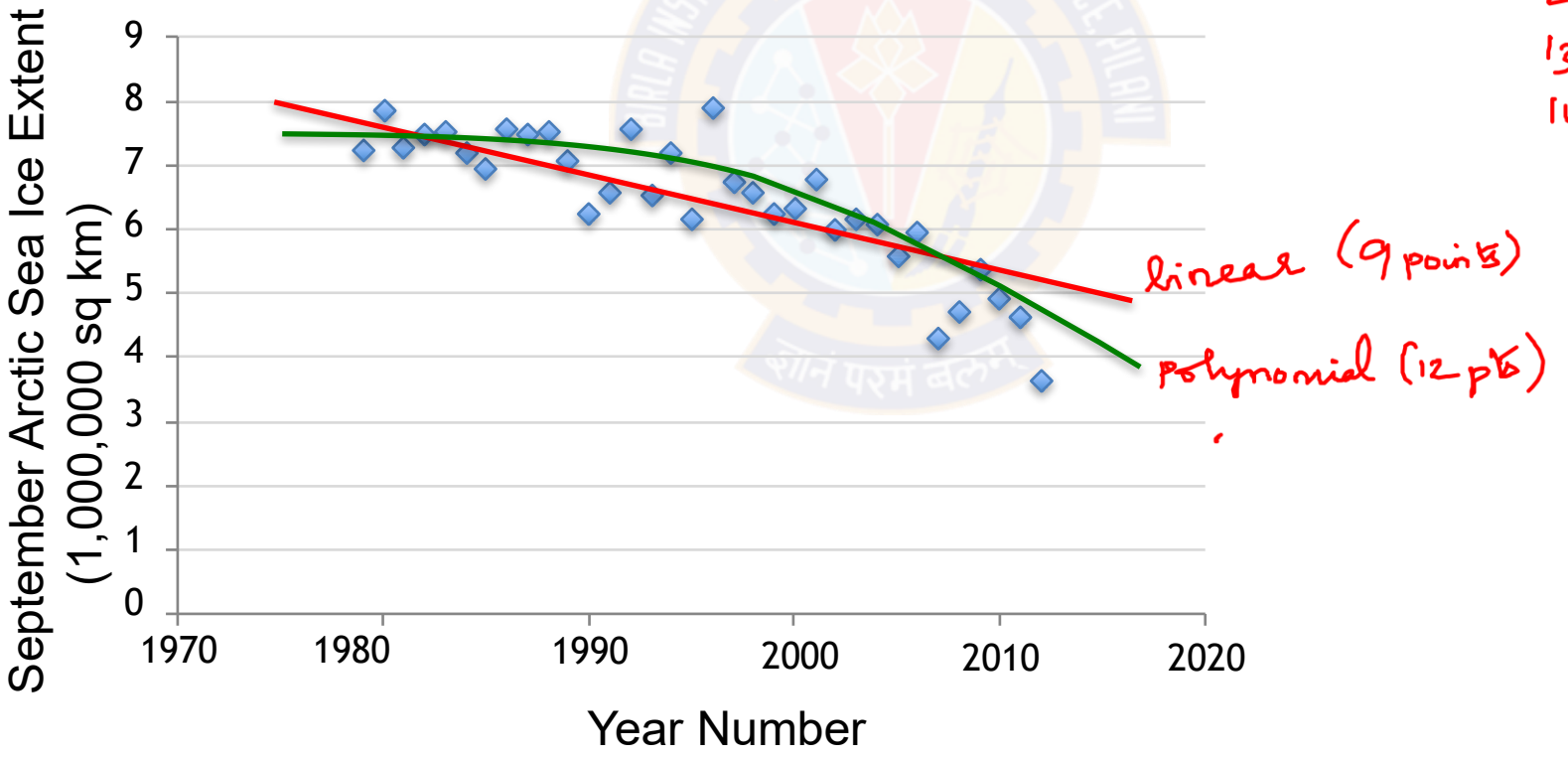
Formal Representation: Interpretation

Supervised Learning: Regression

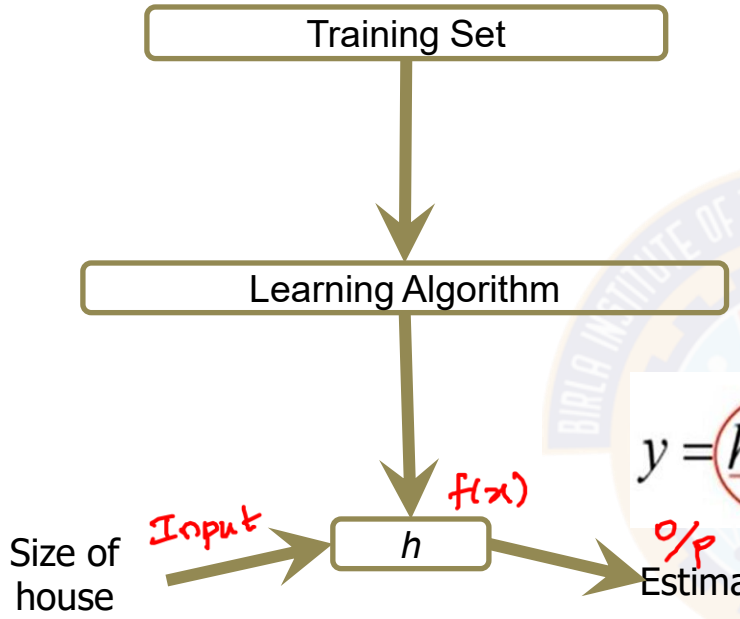
GOAL : Previously unseen records should be assigned a value as accurately as possible.

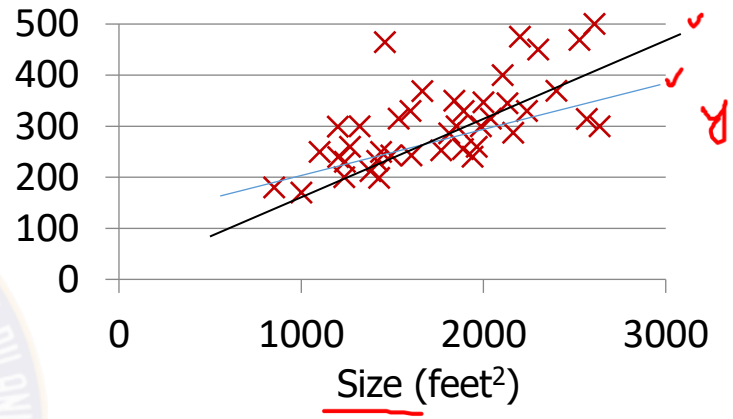
- Given $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
- Learn a function $f(x)$ to predict y given x
  - y is real-valued

x	y
11	3
12	2
13	4
14	?



Machine Learning Process : Interpretation





y-intercept

$y = mx + c$

slope

Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

simple linear regression model

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

θ_i's : Parameters

h(.) : Model

How to choose θ_i's ?

x	y
1	2
2	4
3	6
4	24

x	y
1	1
2	4
3	9
4	16

$y = 2x$ → linear ✓

$y = x^2$ → non-linear

Types of Regression Models

Simple Linear regression model

- (Education) x ⟶ y (Income)

Multiple Linear regression model

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x,$$

- (Education) ✓ x_1
- (Soft Skills) ✓ x_2
- (Experience) ✓ x_3
- (Age) ✓ x_4

→ y (Income)

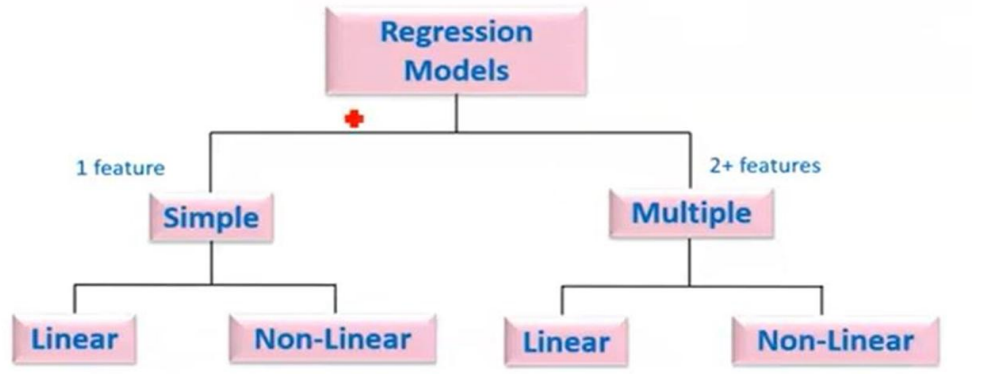
$y = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4$

Multiple Linear regression model ✓

- Hypothesis:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_d x_d = \sum_{j=0}^{d} \theta_j x_j$$

Assume $x_0 = 1$



$y = mx$, $y = 2x$

$y = x^m$, $= x^2$

$y = x_1^2 + 3x_2x_3 + 4x_4^3$

y (pass/fail) = cgpa + iq ✗ (discrete)

House price (y) = sq.ft (x_1) + location (x_2) + base price (x_3)

Electricity consumption = Month + temp + no. of appliances + day

When NOT to use Linear Regression ?

* If the relationship is non-linear
* If the data has outliers, that strongly affects the results
* If the features are highly correlated
* If you are predicting a class / category

Learning the Model Parameters

Analytical

Closed Form Solution Approach

Gradient Descent Approach

How to choose θ_i's ?

Notion of Cost Function

Meal #	Tip amount ($)
1	5.00
2	17.00
3	11.00
4	8.00
5	14.00
6	5.00

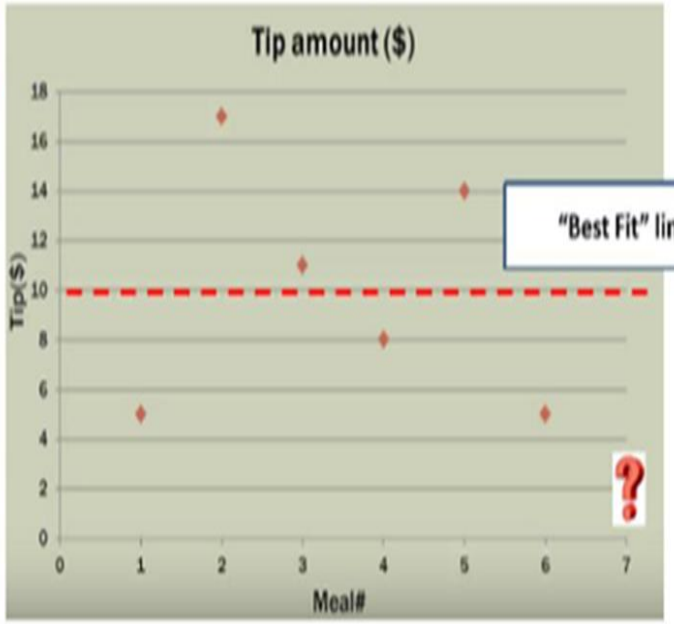
mean = $10 ✓

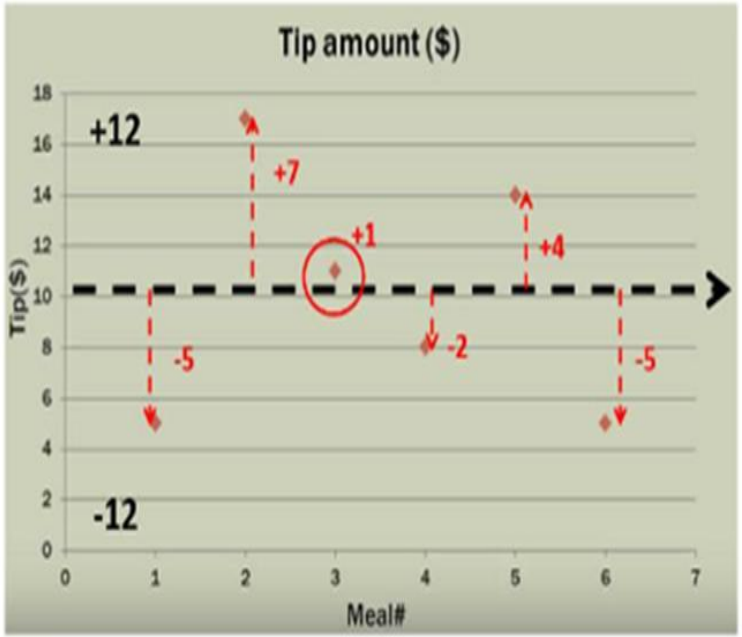
$y = \theta_0$

Notion of Cost Function

Meal #	Tip amount ($)
1	5.00
2	17.00
3	11.00
4	8.00
5	14.00
6	5.00

mean = $10





RESIDULAS (error): Distance from best fit line and the actual values

$y_{act} - y_{obs}$

Meal#	Residual	Residual²
1	−5	25
2	+7	49
3	+1	1
4	−2	4
5	+4	16
6	−5	25

Sum of Squared Errors (SSE) = 120

$0.1^2 = 0.01$

Minimize

Compute θ_1 (slope):

Using least-square method:

$$\theta_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$

$y = \theta_0 + \theta_1 x$

Meal	x_i (Total bill)	y_i (Tip amt)	$x_i - \bar{x}$	$y_i - \bar{y}$	$(x_i - \bar{x})(y_i - \bar{y})$	$(x_i - \bar{x})^2$
1	34	5	-40	-5	200	1600
2	108	17	34	7	238	1156
3	64	11	-10	1	-10	100
4	88	8	14	-2	-28	196
5	99	14	25	4	100	625
6	51	5	-23	-5	115	529

$\bar{x} = 74$ $\bar{y} = 10$

$\sum(x_i - \bar{x})(y_i - \bar{y})$

$$\sum (x_i - \bar{x})(y_i - \bar{y}) = 200 + 238 - 10 - 28 + 100 + 115 = 615$$

$$\sum (x_i - \bar{x})^2 = 1600 + 1156 + 100 + 196 + 625 + 529 = 4206$$

$$\theta_1 = \frac{615}{4206} \approx 0.1462$$ ✓

$y = mx + c$ (c → intercept)

$y = \theta_0 + \theta_1 x$

Compute θ_0 (intercept):

$$\theta_0 = \bar{y} - \theta_1 \bar{x} = 10 - (0.1462 \times 74)$$

$$\theta_0 = 10 - 10.8188 = -0.8188$$

∴ By least-square method

$y = -0.8188 + 0.1462x$ ✓

Notion of Cost Function

Meal #	Tip amount ($)
1	5.00
2	17.00
3	11.00
4	8.00
5	14.00
6	5.00

mean = $10

Meal	Total bill ($)	Tip amount ($)	$\hat{y}_i = 0.1462x - 0.8188$	$\hat{y}_i$ (predicted tip amount)
	x	y		
1	34	5	$\hat{y}_i = 0.1462(34) - 0.8188$	4.1505
2	108	17	$\hat{y}_i = 0.1462(108) - 0.8188$	14.9693
3	64	11	$\hat{y}_i = 0.1462(64) - 0.8188$	8.5365
4	88	8	$\hat{y}_i = 0.1462(88) - 0.8188$	12.0453
5	99	14	$\hat{y}_i = 0.1462(99) - 0.8188$	13.6535
6	51	5	$\hat{y}_i = 0.1462(51) - 0.8188$	6.6359
	$\bar{x} = 74$	$\bar{y} = 10$	Observed vs. Predicted	

Actual

Predicted

$y = \theta_0 + \theta_1 x$

$y = mx + c$

Sum of squared Errors

$$J(\theta) = \frac{1}{2n}\sum_{i=1}^{n}(\hat{y}^{(i)} - y^{(i)})^2$$

$$J(\theta) = \frac{1}{2n}\sum_{i=1}^{n}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$(y_{act} - y_{obs})^2$

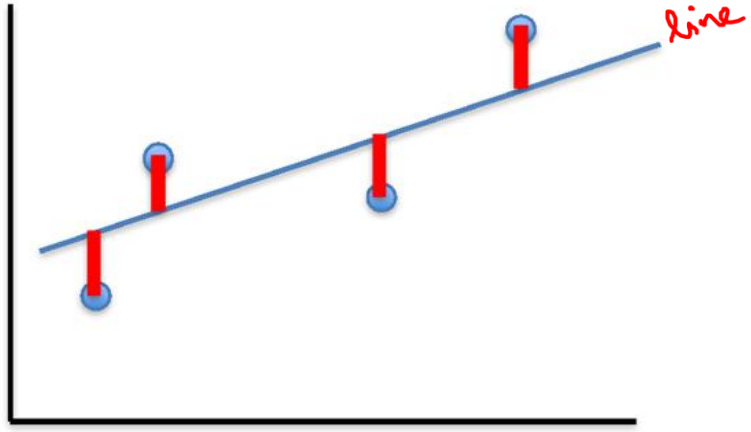
$\hat{y}_i = h_\theta(x^i)$

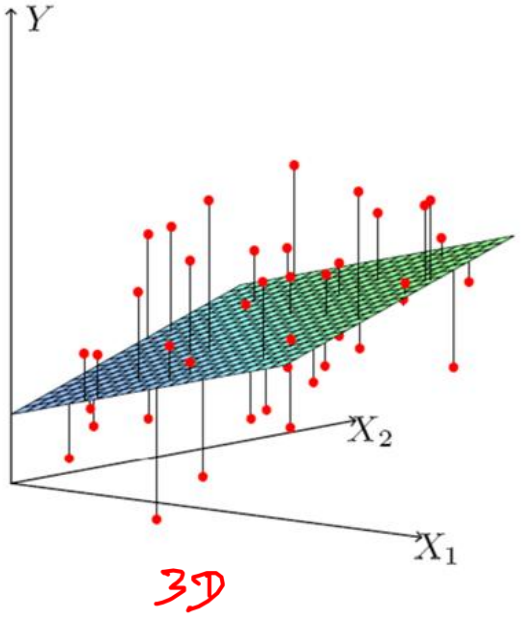
Linear Regression : Least Squares

- Cost Function

$$J(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2$$

- Fit by solving $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$





- Hypothesis $h_\theta(x) = \theta_0 + \theta_1 x$
- Parameters θ_0 and θ_1
- Cost Function $J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^i) - y^{(i)})^2$
- Goal $\underset{\theta_0,\theta_1}{\text{minimize}}\ J(\theta_0,\theta_1)$

Intuition Behind Cost Function

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)^2$$

For insight on J(), let's assume $x \in \mathbb{R}$ so $\boldsymbol{\theta} = [\theta_0, \theta_1]$

Source Credit : Based on example by Andrew Ng

Intuition Behind Cost Function

$$J(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2$$

$\theta_0 \rightarrow$ y-intercept

$h_\theta(x) = \theta_0 + \theta_1 x$

when $\theta_1 = 1$

$\therefore h_\theta(x) = \theta_0 + x$

For insight on J(), let's assume $x \in \mathbb{R}$ so $\boldsymbol{\theta} = [\theta_0, \theta_1]$

$h_\theta(x) = x$

when $\theta_0 = 0$ & $\theta_1 = 1$

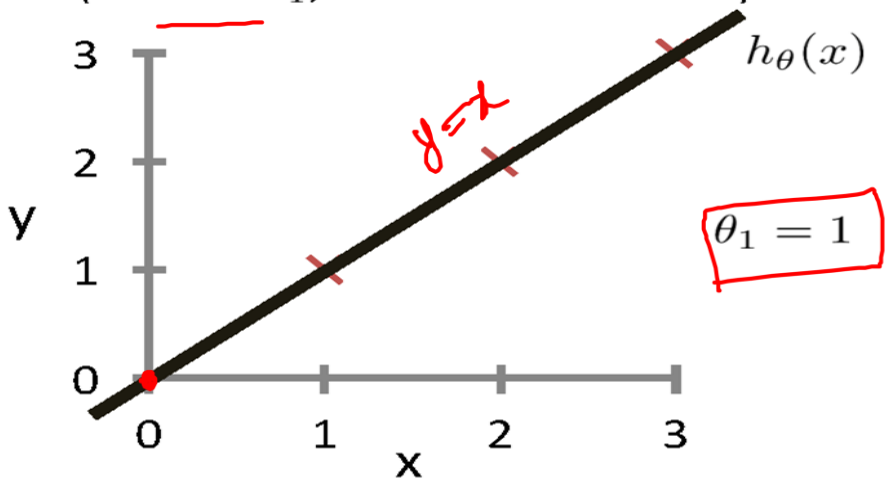
$\theta_0 = 0$

$h_\theta(x) = \theta_1 x$

$y = mx$

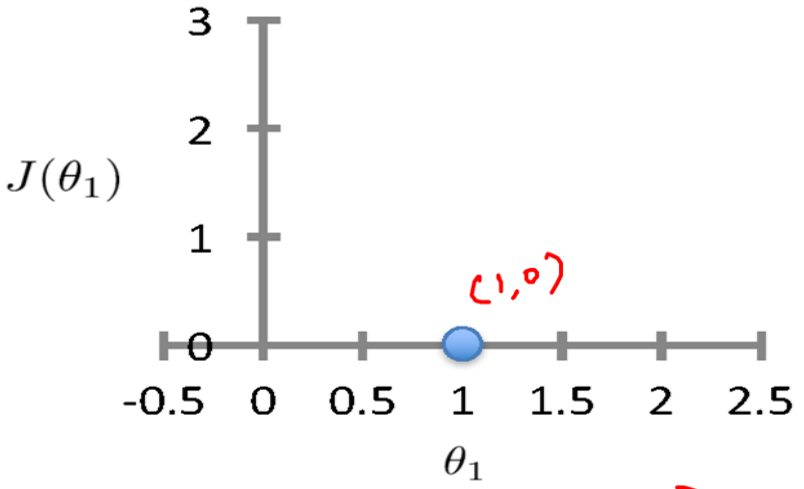
$h_\theta(x)$

(for fixed θ_1, this is a function of x)



$J(\theta_1)$

(function of the parameter θ_1)



$x = 0 \rightarrow$ y-axis

$y = 0 \rightarrow$ x-axis

Source Credit : Based on example by Andrew Ng

Intuition Behind Cost Function

$$J(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2$$

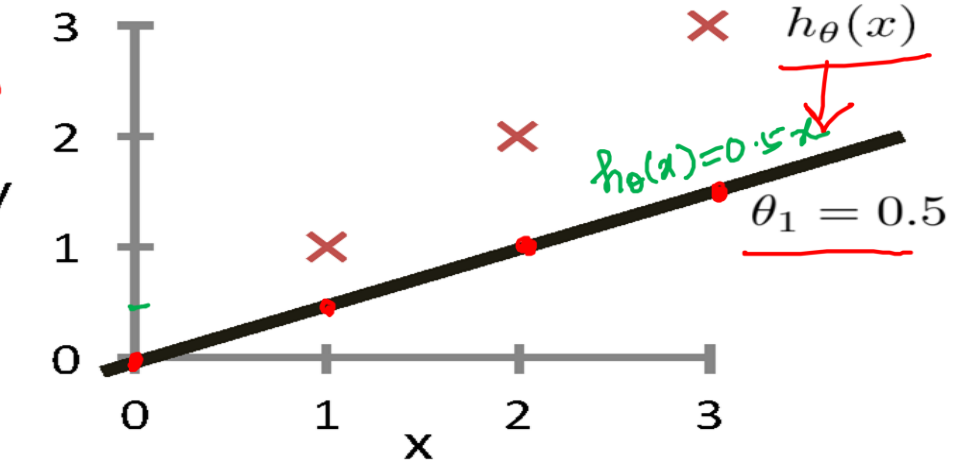
$h_\theta(x) = \theta_0 + \theta_1 x$ so you know

For insight on J(), let's assume $x \in \mathbb{R}$ so $\boldsymbol{\theta} = [\theta_0, \theta_1]$ $\theta_0 = 0$, $\theta_1 = 0.5$

x	y
1	2
2	3
3	3

$h_\theta(x)$

(for fixed θ_1, this is a function of x)

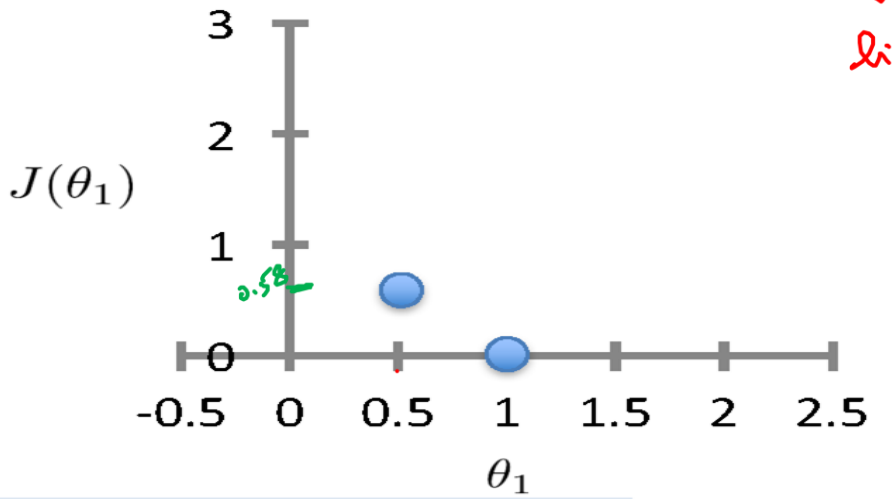


$h_\theta(x) = x/2$

x	
1	1/2
2	1
3	3/2

$J(\theta_1)$

(function of the parameter θ_1)



$\therefore h_\theta(x) = 0.5x = x/2$ → line

Based on example by Andrew Ng

$$J([0, 0.5]) = \frac{1}{2 \times 3}\left[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2\right] \approx 0.58$$

Assume $\theta_0 = 0$, $\theta_1 = 0.5$

Eg:

x_i	y_i	$h_\theta(x) = 0.5x$	$h_\theta(x) = x$	$h_\theta(x_i) - y_i$
1	1	$h_\theta(1) = 0.5$	1	0.5 − 1
2	2	$h_\theta(2) = 0.5(2) = 1$	2	1 − 2
3	3	$h_\theta(3) = 0.5(3) = 1.5$	3	1.5 − 3

Compute $J(\theta)$ for $\theta = [\underset{\theta_0}{0}, \underset{\theta_1}{0.5}] \Rightarrow h_\theta(x) = 0.5x$

$$J(\theta) = \frac{1}{2n} \sum_{i=1}^{n} \left[h_\theta(x_i) - y_i \right]^2$$ Sum of squared errors

WKT $n = 3$ ← no. of data

$$= \frac{1}{2(3)} \left[\underset{x=1}{(0.5 - 1)^2} + \underset{x=2}{(1 - 2)^2} + \underset{x=3}{(1.5 - 3)^2} \right] = 3.5/6 \approx 0.58$$

(ii) let $\theta = [0, 1] \Rightarrow h_\theta(x) = x$

$\theta_1 = 1$ $\quad J(\theta) = J(1) = \frac{1}{6} \left[(1-1)^2 + (2-2)^2 + (3-3)^2 \right] = 0$

Try

$\theta_0 = 0$

when $\theta_1 = 1.5$ $\quad h_\theta(x) = 1.5x$

x	$h_\theta(x) = 1.5x$	$h_\theta(x) = 2x$
1	1.5	2
2	2(1.5) = 3	4
3	3(1.5) = 4.5	6

$$J(\theta_1) = J(1.5) = \frac{1}{6}\left[(1.5-1)^2 + (3-2)^2 + (4.5-3)^2\right]$$

$$= 0.583$$

when $\theta_1 = 2$ $\quad h_\theta(x) = 2x$

$$J(2) = \frac{1}{6}\left[(2-1)^2 + (4-2)^2 + (6-3)^2\right] = 2.33$$

Intuition Behind Cost Function

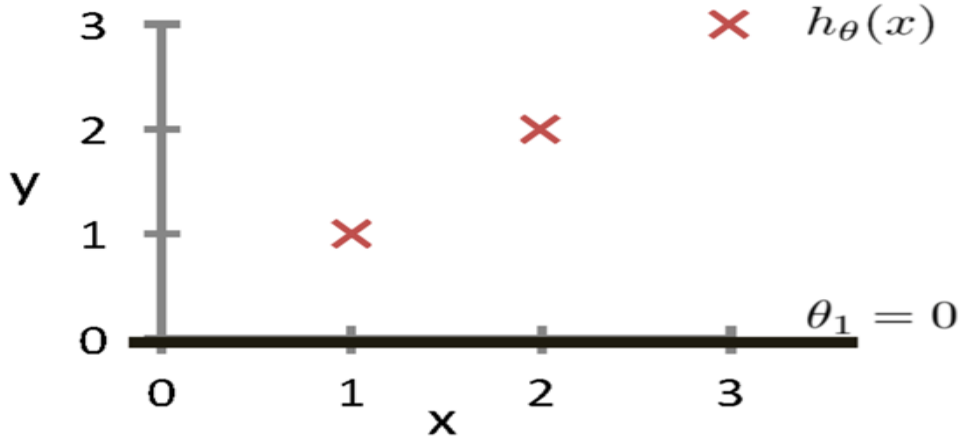
$$J(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2$$

$h_\theta(x) = \theta_0 + \theta_1 x$

For insight on J(), let's assume $x \in \mathbb{R}$ so $\boldsymbol{\theta} = [\theta_0, \theta_1]$

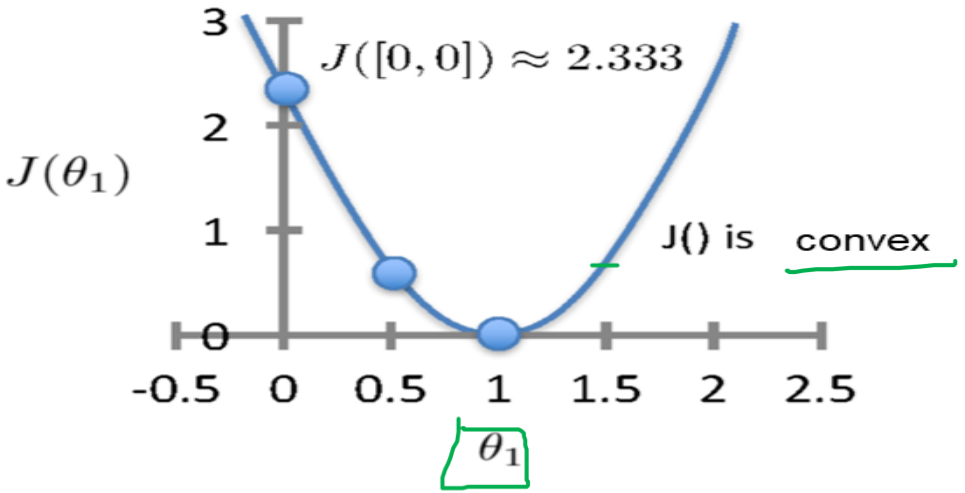
$h_\theta(x)$

(for fixed θ_1, this is a function of x)



$J(\theta_1)$

(function of the parameter θ_1)



$\theta_0 = 0, \theta_1 = 1$

Best fit line = $h_\theta(x) = x$

Based on example by Andrew Ng

Source Credit : by Andrew Ng

Notion of Maxima and Minima of a function - In General

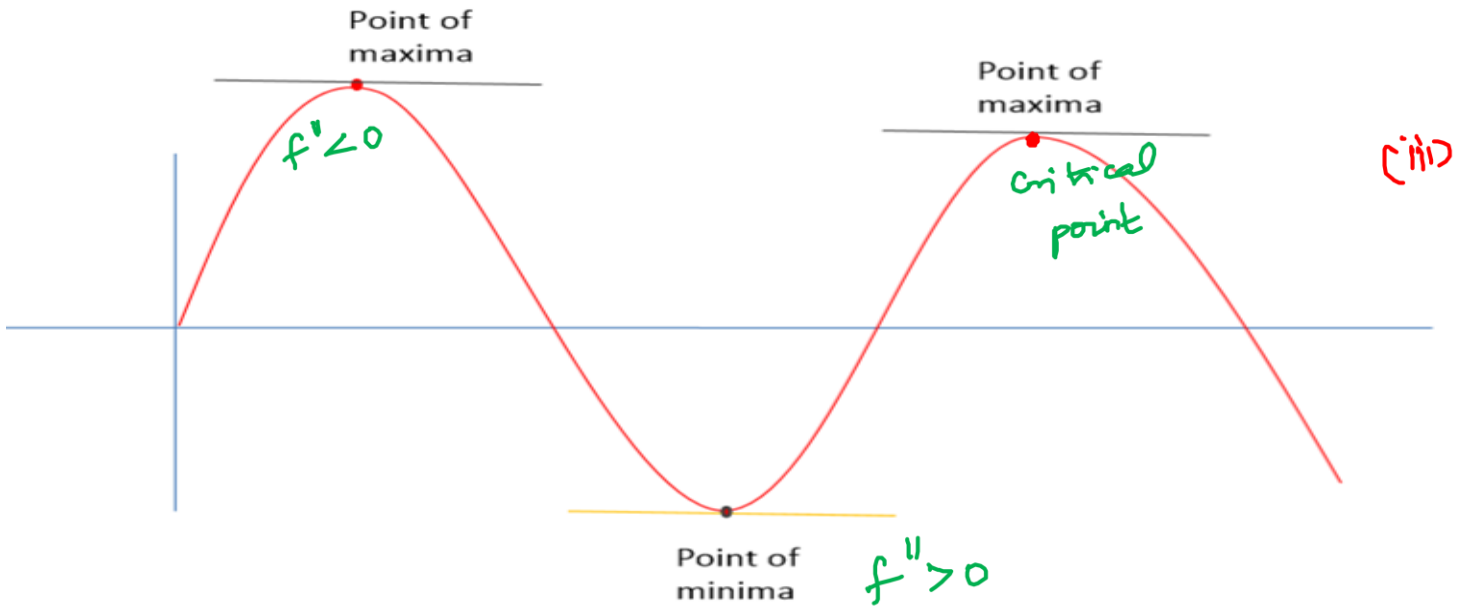
$m = \frac{y_2 - y_1}{x_2 - x_1}$

(x_1, y_1) (x_2, y_2)

(1, 2) (3, 4)

$y = mx + C$ (m: slope)

- For maxima and minima m=dy/dx=tan 0 ^{0}=0
- dy/dx = 0 means tangent is parralel to X –axis.



(i) Step1 : Find derivative

(ii) Step2: Find the critical pts

$\frac{df}{dx} = 0$

(iii) Step3:

$\frac{d^2f}{dx^2} < 0 \rightarrow$ Maximum

$\frac{d^2f}{dx^2} > 0 \rightarrow$ Minimum

$f(x) = x^2 - 4x + 3$

x	$y = f(x)$
1	
−1	
2	
3	
5	

Eg: $f(x) = x^2 - 4x + 3$ Find the pt of maxima/minima

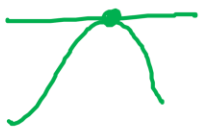
Ans:

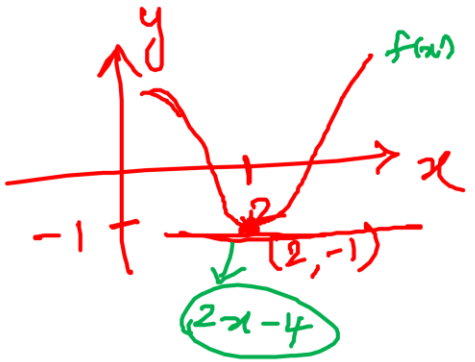
Step 1: $\frac{df}{dx} = f'(x) = 2x - 4$

Step 2: $f'(x) = 0$ find the critical points

$\therefore 2x - 4 = 0$

$\Rightarrow \boxed{x = 2}$





Step 3: Find the second derivative

$f''(x) = 2$

$\frac{d^2f}{dx^2} = f''(x) = \frac{d}{dx}(2x-4) = 2 > 0$ $\therefore$ The function is minimum at $x = 2$

What is the value of the fn. at $x = 2$?

Sub. $x = 2$ in $f(x) = 2^2 - 4(2) + 3 = -1$

Suppose $f''(x) = -2$

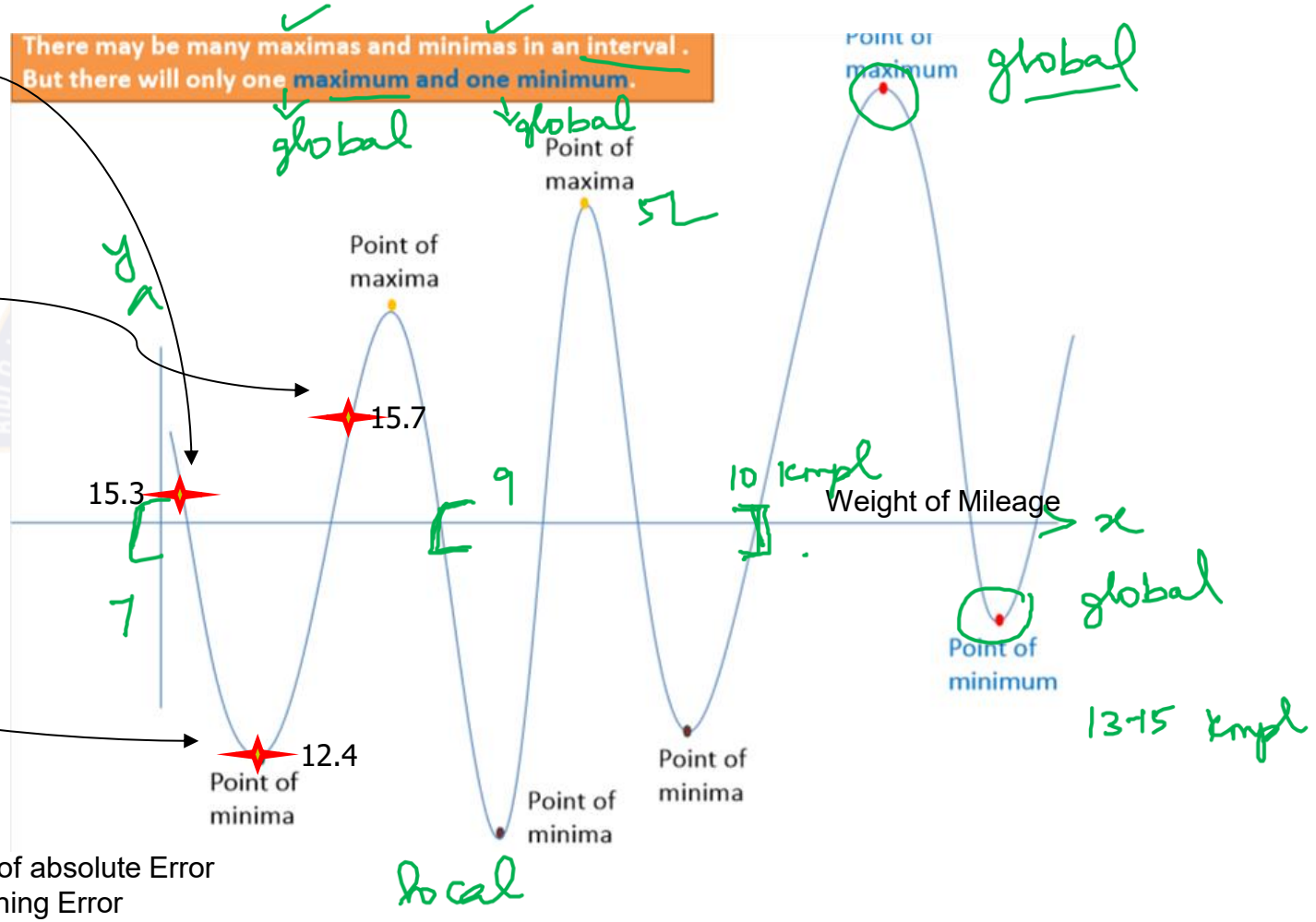
Then $f(x)$ will have maximum

Notion of Maxima and Minima of a function - In Machine Learning

CarPrice = 1 + 0.05Mileage

CarPrice = 1 + 0.25Mileage

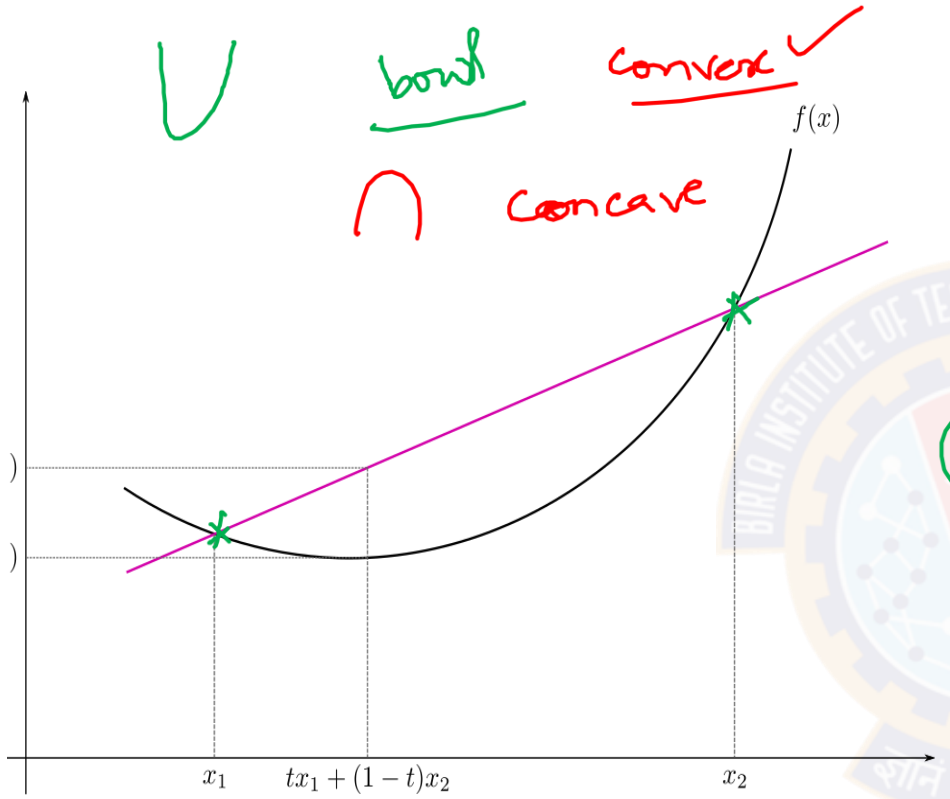
CarPrice = 1 + 0.75Mileage

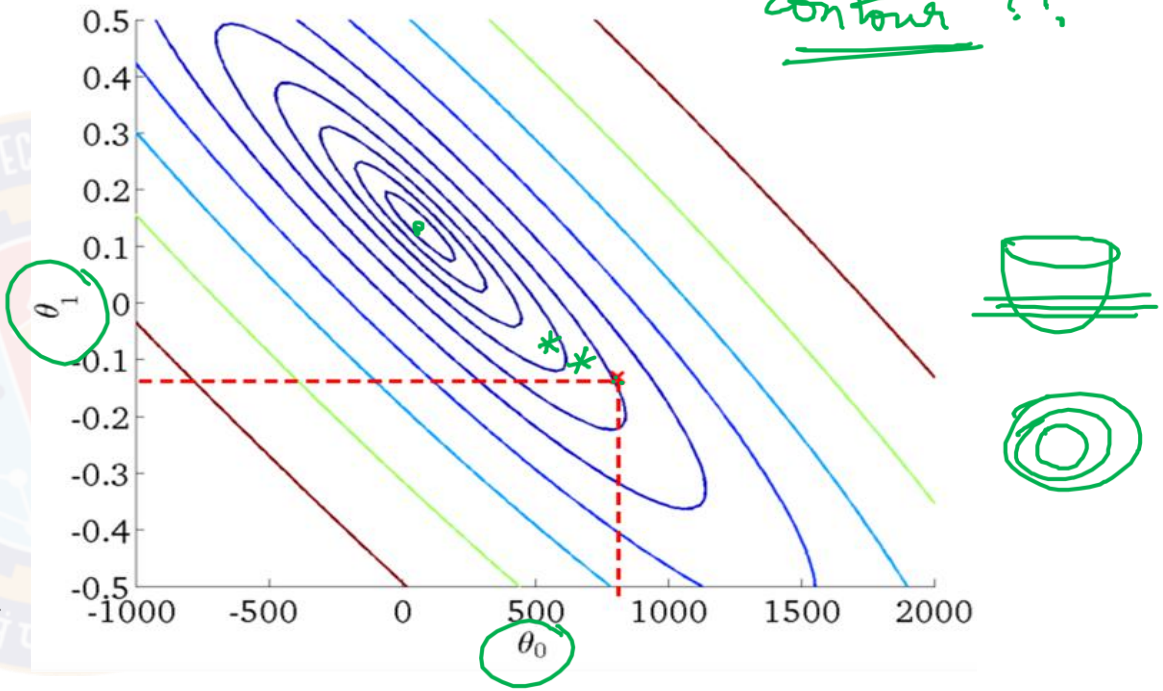


Mileage (in kmpl)	*Car Price (in cr)*
9.8	10.48
9.12	1.75
9.5	6.95
10	2.51

Sum of absolute Error / Training Error

Convex function: Multivariate View





[Real-valued function] defined on an [*n*-dimensional interval] is called **convex** if the [line segment] between any two points on the [graph of the function] lies above or on the graph

Closed Form Solution

Mathematical formula to predict the values

Vectorization

- Benefits of vectorization
  - More compact equations
  - Faster code (using optimized matrix libraries)
- Consider our model:

$$h(\boldsymbol{x}) = \sum_{j=0}^{d} \theta_j x_j$$

$= \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \dots \theta_d x_d$

$x_0 = 1$

- Let

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \qquad \boldsymbol{x}^\intercal = \begin{bmatrix} 1 & x_1 & \dots & x_d \end{bmatrix}$$

- Can write the model in vectorized form as $h(\boldsymbol{x}) = \boldsymbol{\theta}^\intercal \boldsymbol{x}$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \quad x^T = \begin{bmatrix} 1 & No.of.Years\ Experience \end{bmatrix}$$

X No.of.Years of Experience (in Years)	Y Salary Of the Employee (in Lakhs)
1	2
2	3
3	4
4	5
5	6

$y = mx + c$

$\theta_0 + \theta_1 x$

$a = [1, 2]^T = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ 2x1

$y = \theta_1 x + \theta_0$

Vectorization

X No.of.Years of Experience (in Years)	Y Salary Of the Employee (in Lakhs)
1	2
2	3
3	4
4	5
5	6

- Consider our model for n instances:

$$h\left(\boldsymbol{x}^{(i)}\right) = \sum_{j=0}^{d} \theta_j x_j^{(i)}$$

- Let

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \quad \boldsymbol{X} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(i)} & \dots & x_d^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix}$$

column vector

$\mathbb{R}^{(d+1)\times 1}$ $\quad$ $\mathbb{R}^{n\times(d+1)}$

Let:

$$\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

$n \times 1$

- Can write the model in vectorized form as $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{X}\boldsymbol{\theta}$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \quad X = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix} \quad y = \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$$

$X\theta = n \times (d+1) \cdot (d+1) \times 1 = n \times 1$

Vectorization

- For the linear regression cost function:

$$J(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2$$

$$= \frac{1}{2n}\sum_{i=1}^{n}\left(\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}^{(i)} - y^{(i)}\right)^2$$

$$\sum_{i=1}^{n} v^2 = v^T v \quad \text{for any vector, } v$$

Let:

$$\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

$$J(\theta) = \frac{1}{2n}\underbrace{(\boldsymbol{X\theta} - \boldsymbol{y})^{\mathsf{T}}}_{\mathbb{R}^{1\times n}}\underbrace{(\boldsymbol{X\theta} - \boldsymbol{y})}_{\mathbb{R}^{n\times 1}}$$

$\boldsymbol{X}$: $\mathbb{R}^{n\times(d+1)}$, $\boldsymbol{\theta}$: $\mathbb{R}^{(d+1)\times 1}$

since $\theta^T x = X\theta$

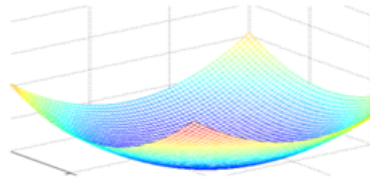
$n\times 1$

X No.of.Years of Experience (in Years)	Y Salary Of the Employee (in Lakhs)
1	2
2	3
3	4
4	5
5	6

$$\theta^T = \begin{bmatrix} \theta_0 & \theta_1 \end{bmatrix} \quad X^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} \quad y = \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$$

Vectorization

- Instead of using GD, solve for optimal $\boldsymbol{\theta}$ analytically
  - Notice that the solution is when $\frac{\partial}{\partial \boldsymbol{\theta}} J(\boldsymbol{\theta}) = 0$



- Derivation:

$$\mathcal{J}(\boldsymbol{\theta}) = \frac{1}{2n}(\boldsymbol{X\theta} - \boldsymbol{y})^\intercal (\boldsymbol{X\theta} - \boldsymbol{y})$$

1 x 1 (scalar)

$$\propto \boldsymbol{\theta}^\intercal \boldsymbol{X}^\intercal \boldsymbol{X\theta} - \boldsymbol{y}^\intercal \boldsymbol{X\theta} - \boldsymbol{\theta}^\intercal \boldsymbol{X}^\intercal \boldsymbol{y} + \boldsymbol{y}^\intercal \boldsymbol{y}$$

$$\propto \boldsymbol{\theta}^\intercal \boldsymbol{X}^\intercal \boldsymbol{X\theta} - 2\boldsymbol{\theta}^\intercal \boldsymbol{X}^\intercal \boldsymbol{y} + \boldsymbol{y}^\intercal \boldsymbol{y}$$

Take derivative and set equal to 0, then solve for $\boldsymbol{\theta}$:

$$\frac{\partial}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}^\intercal \boldsymbol{X}^\intercal \boldsymbol{X\theta} - 2\boldsymbol{\theta}^\intercal \boldsymbol{X}^\intercal \boldsymbol{y} + \boldsymbol{y}^\intercal \boldsymbol{y}) = 0$$

$$(\boldsymbol{X}^\intercal \boldsymbol{X})\boldsymbol{\theta} - \boldsymbol{X}^\intercal \boldsymbol{y} = 0$$

$$(\boldsymbol{X}^\intercal \boldsymbol{X})\boldsymbol{\theta} = \boldsymbol{X}^\intercal \boldsymbol{y}$$

$$\theta = \frac{X^T y}{X^T X}$$

$p = \frac{a}{b}$

$p = ab^{-1}$

Closed Form Solution: $\boldsymbol{\theta} = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal \boldsymbol{y}$

for large n' computationally expensive

invertible

Simplification

$$J(\theta) = \frac{1}{2n}\left[(X\theta - y)^T (X\theta - y)\right]$$

$\frac{d}{d\theta}$ → single variable

$\frac{\partial}{\partial\theta}$ → multiple variables

$$= \frac{1}{2n}\left[(X\theta)^T - y^T)(X\theta - y)\right]$$

$$= \frac{1}{2n}\left[(\theta^T X^T - y^T)(X\theta - y)\right]$$

$$= \frac{1}{2n}\left[\theta^T X^T X\theta - y\theta^T X^T - y^T X\theta + yy^T\right]$$

scalars & equal

$$J(\theta) = \frac{1}{2n}\left[\theta^T X^T X\theta - 2\theta^T X^T y + yy^T\right]$$

$$\frac{\partial}{\partial\theta} J(\theta) = \frac{1}{2n}\left[2X^T X\theta - 2X^T y\right]$$

$$\frac{\partial}{\partial\theta}[J(\theta)] = \frac{2}{2n}\left[X^T X\theta - X^T y\right]$$

$$(a-b)^T = a^T - b^T$$

$$(ab)^T = b^T a^T$$

$$\theta^T X^T y = (y^T X\theta)^T$$

$$= yX^T\theta^T$$

Note.

$$\frac{\partial}{\partial\theta}(c) = 0$$

$$\frac{\partial}{\partial\theta}(b^T\theta) = b$$

$$\frac{\partial}{\partial\theta}(\theta^T A\theta) = 2A\theta$$

(If A is symmetric)

Vectorization

- Can obtain $\boldsymbol{\theta}$ by simply plugging $\boldsymbol{X}$ and $\boldsymbol{y}$ into

$$\boldsymbol{\theta} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y}$$

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(i)} & \dots & x_d^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix} \quad \boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

$n \times (d+1)$

$C_3 \rightarrow 2C_2$

C_1	C_2	C_3
1	x	$2x$
1	$2x$	$4x$
1	$3x$	$6x$

linearly dependent

- If $\boldsymbol{X}^\top \boldsymbol{X}$ is not invertible (i.e., singular), may need to:
  - Use pseudo-inverse instead of the inverse
    - In python, `numpy.linalg.pinv(a)`
  - Remove redundant (not linearly independent) features
  - Remove extra features to ensure that $d \leq n$

Fit the Linear Regression Model :

Using Closed Form – Problem Type 1

$(X^TX)^{-1}(X^Ty) = \theta$

$$\left(X^T \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} * X \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix} \right)^{-1} * X^T \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} * y \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$$

2×5 , 5×2

X No.of.Years of Experience (in Years)	Y Salary Of the Employee (in Lakhs)
1	2
2	3
3	4
4	5
5	6

$$\left(\begin{bmatrix} 5 & 15 \\ 15 & 55 \end{bmatrix} \right)^{-1} * X^T \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} * y \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$$

find inverse of this matrix

$$\left(\begin{bmatrix} 1.1 & -0.3 \\ -0.3 & 0.1 \end{bmatrix} \right) * X^T \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} * y \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$$

inverse of matrix

Can obtain θ by simply plugging X and y into

$$\theta = (X^TX)^{-1}X^Ty$$

$$\begin{bmatrix} 0.8 & 0.5 & 0.2 & -0.1 & -0.4 \\ -0.2 & -0.1 & 0 & 0.1 & -0.2 \end{bmatrix} * y \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$$

$$\theta^T = \begin{bmatrix} \theta_0 & \theta_1 \end{bmatrix} \quad X = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} \quad y = \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$$

$$\theta = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$ ✓

$\theta_0 = 1$, $\theta_1 = 1$

$h_\theta(x) = 1 + x$ ✓

Gradient Descent Approach

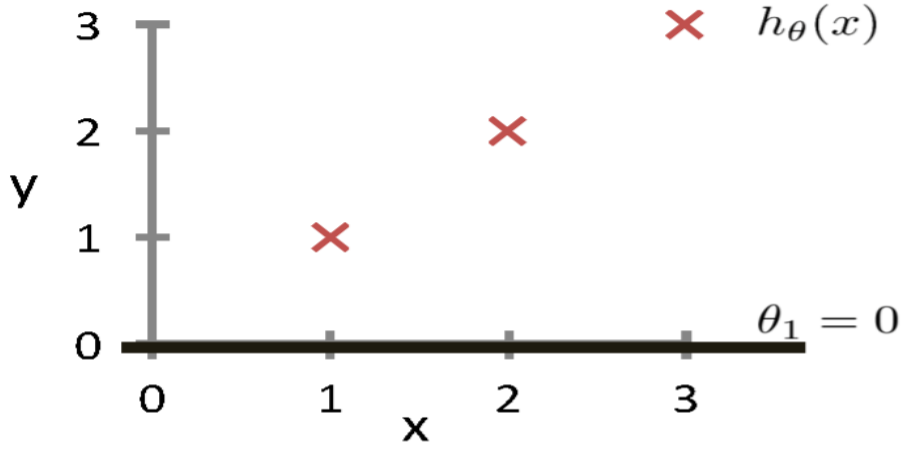
Intuition Behind Cost Function

$$J(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2$$

For insight on J(), let's assume $x \in \mathbb{R}$ so $\boldsymbol{\theta} = [\theta_0, \theta_1]$

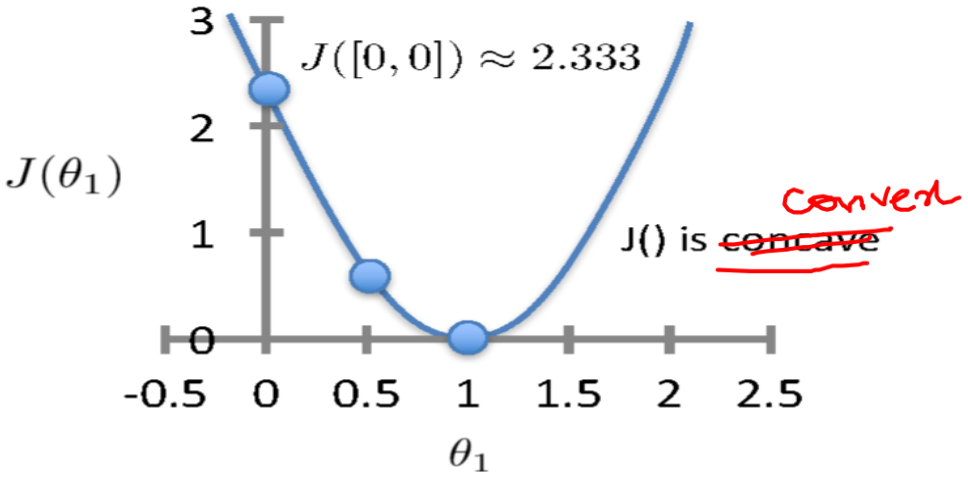
$h_\theta(x)$

(for fixed θ_1, this is a function of x)



$J(\theta_1)$

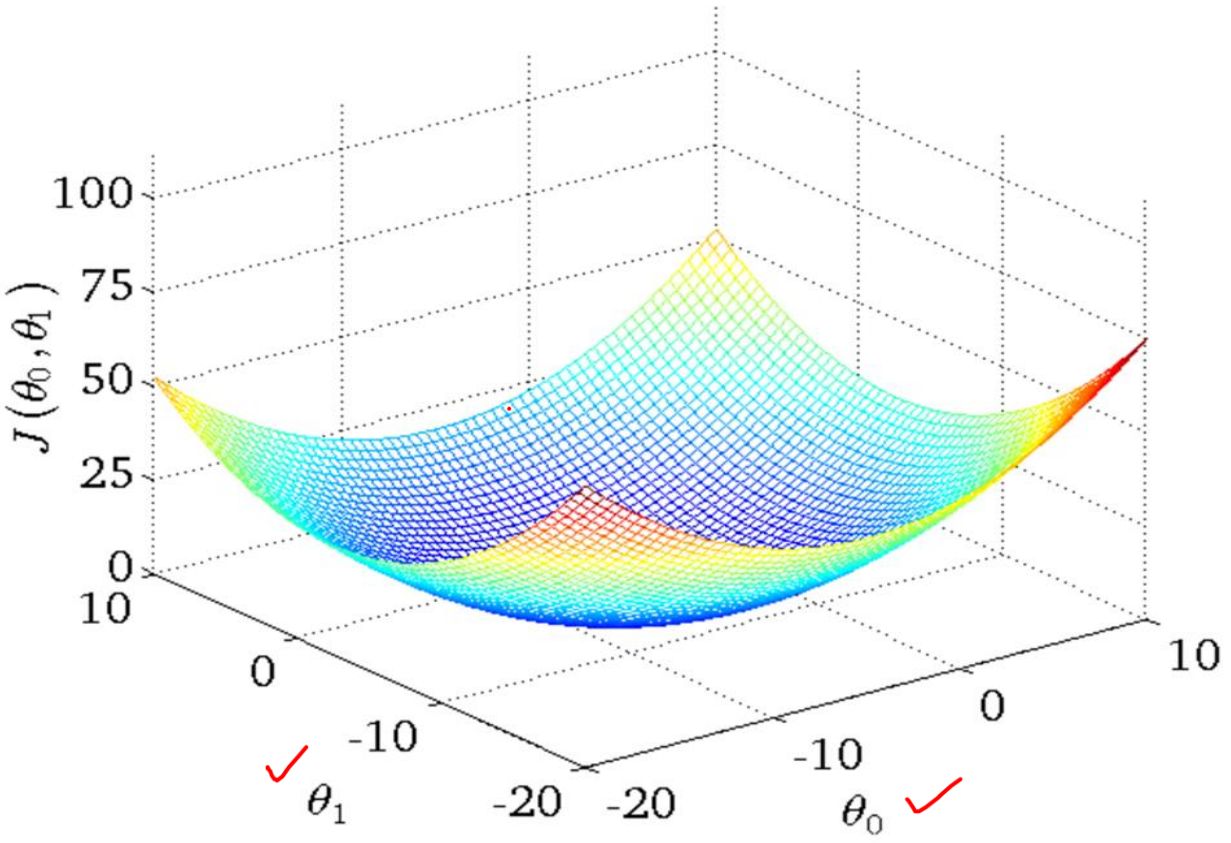
(function of the parameter θ_1)



2D

Source Credit : Based on example by Andrew Ng

Intuition Behind Cost Function



Based on example by Andrew Ng

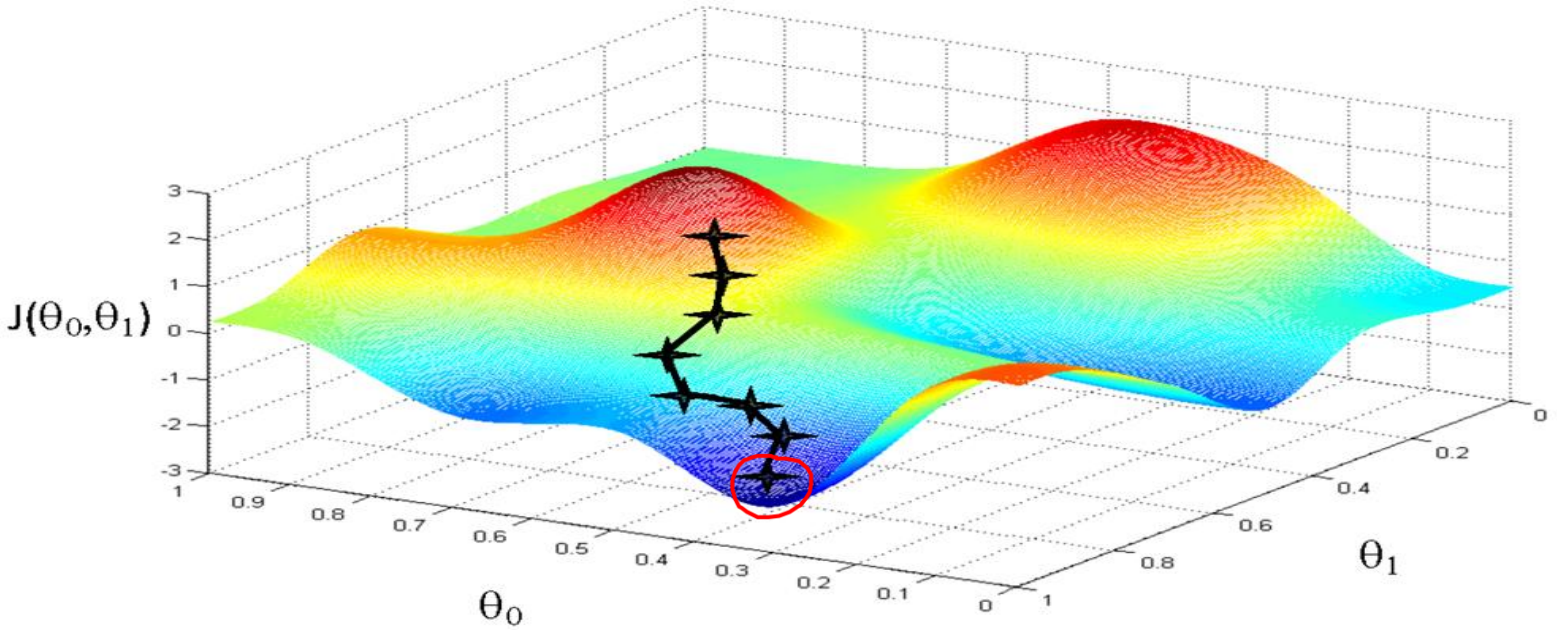
Source Credit :

Basic Search Procedure

- Choose initial value for θ
- Until we reach a minimum:
  - Choose a new value for θ to reduce $J(\theta)$

$\theta_0 = 0$, $\theta_1 = 1$

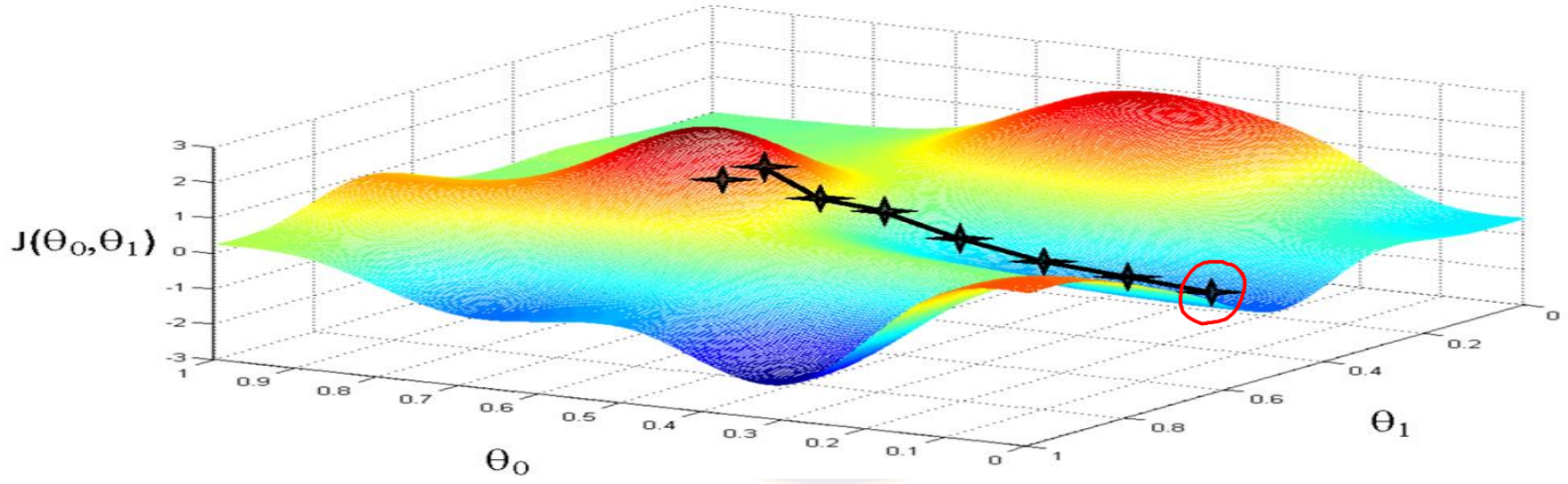
$\theta_1 = 0.5$



Source Credit : Figure by Andrew Ng

Basic Search Procedure

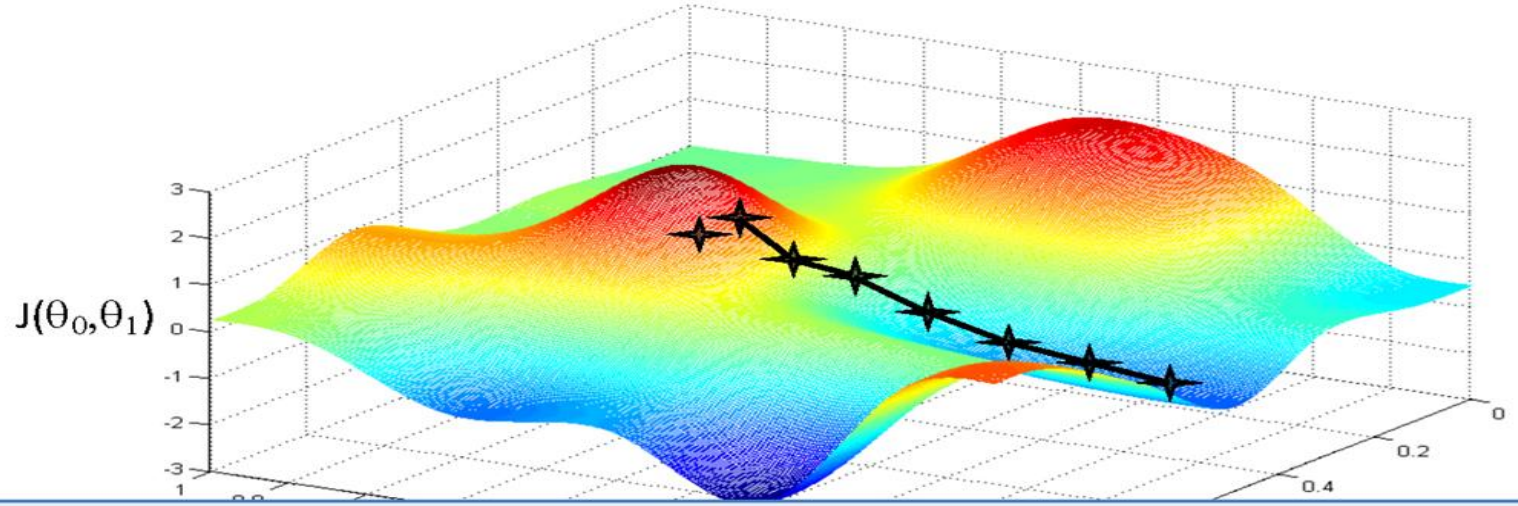
- Choose initial value for θ
- Until we reach a minimum:
  - Choose a new value for θ to reduce $J(\theta)$



Source Credit : Figure by Andrew Ng

Basic Search Procedure

- Choose initial value for θ
- Until we reach a minimum:
  - Choose a new value for θ to reduce $J(\theta)$



Since the least squares objective function is convex (concave), we don't need to worry about local minima

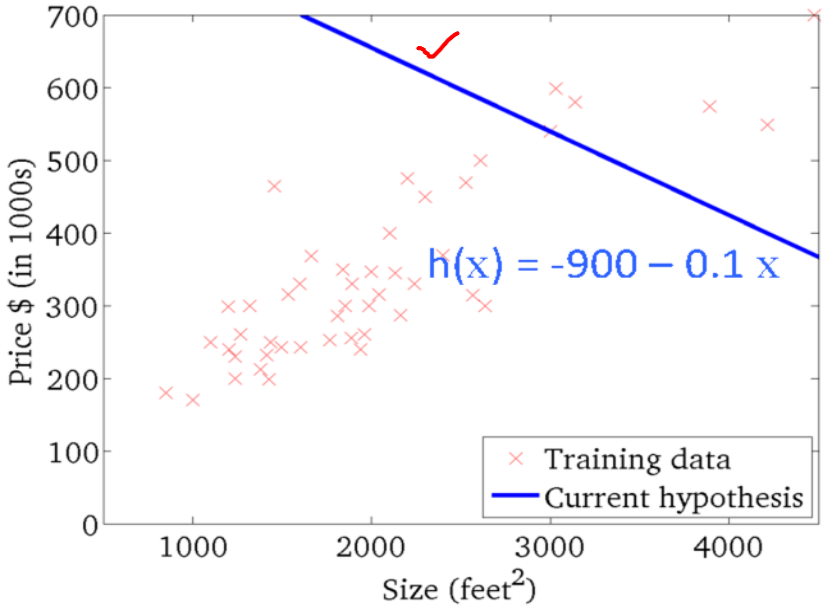
Source Credit : Figure by Andrew Ng

Gradient Descent

Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

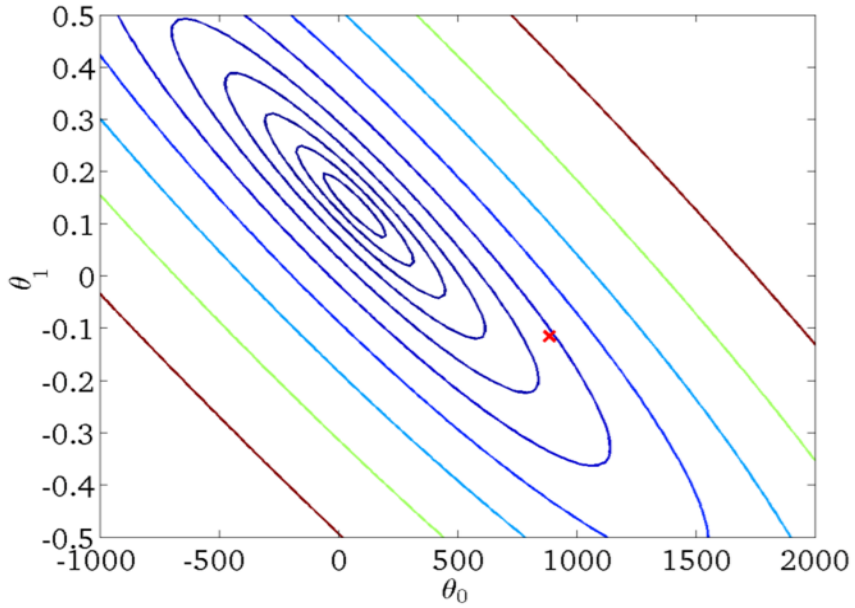
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

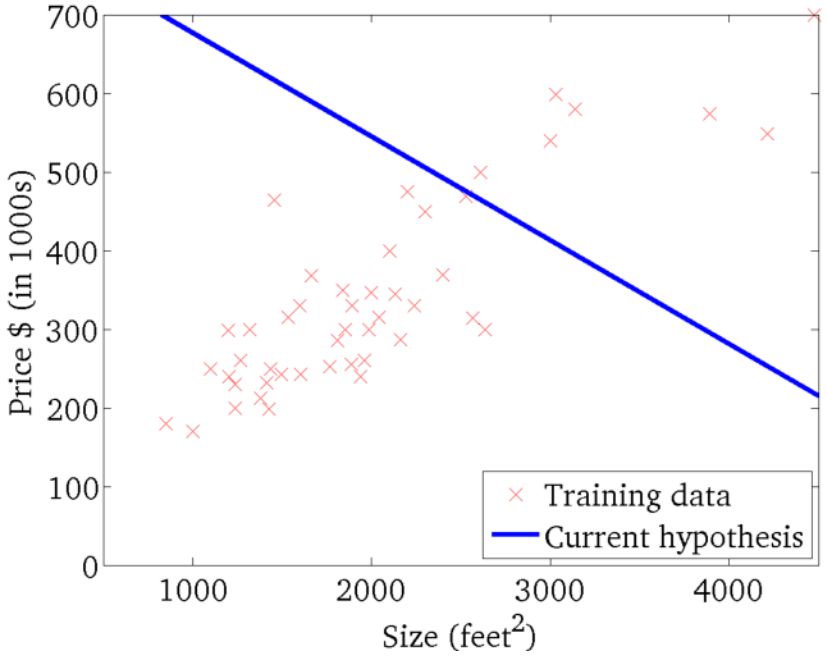


Source Credit : Slide by Andrew Ng

Gradient Descent

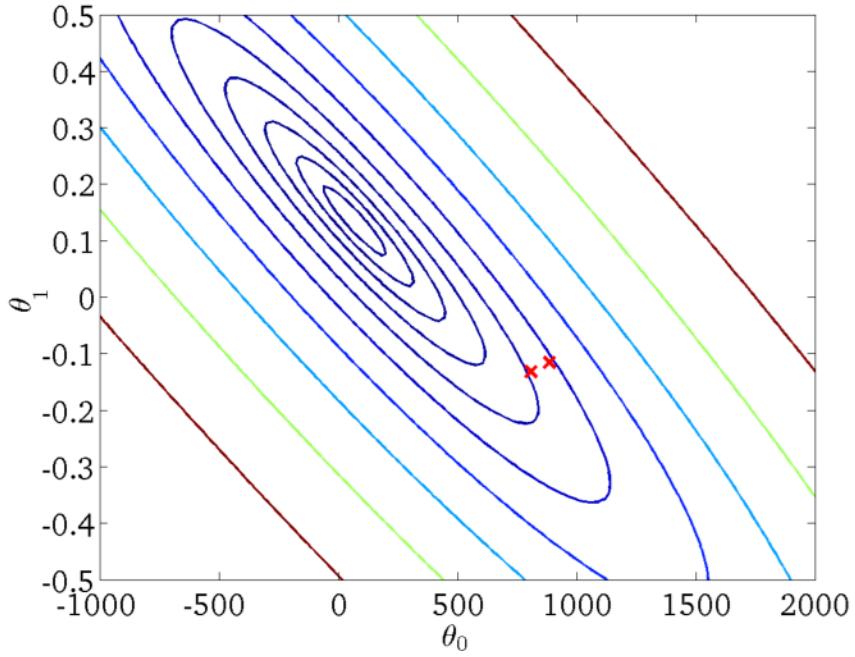
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

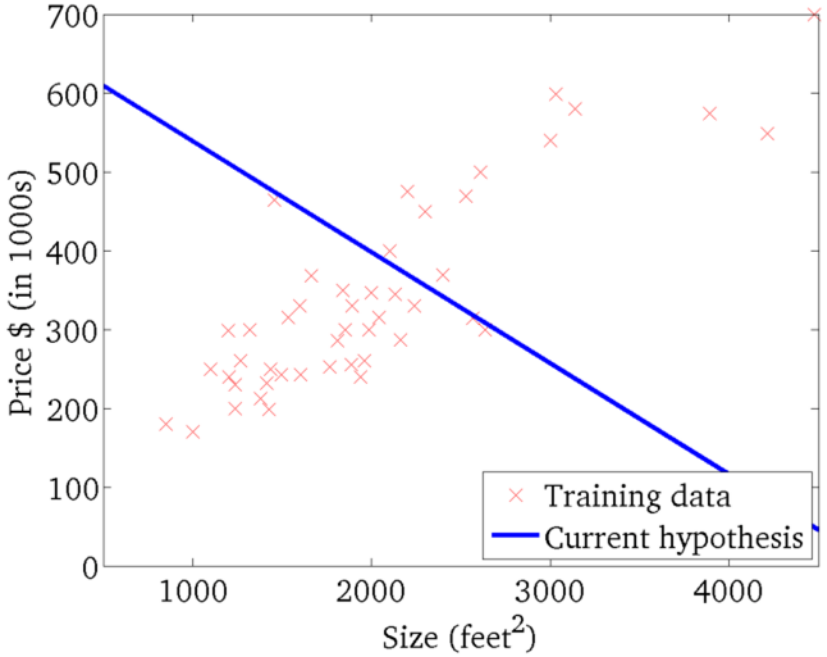


Source Credit : Slide by Andrew Ng

Gradient Descent

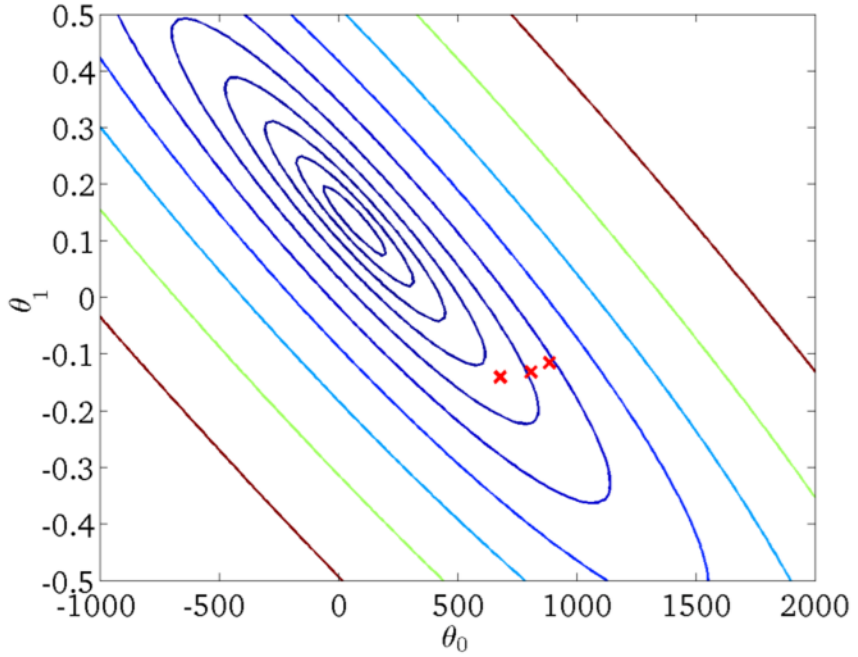
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

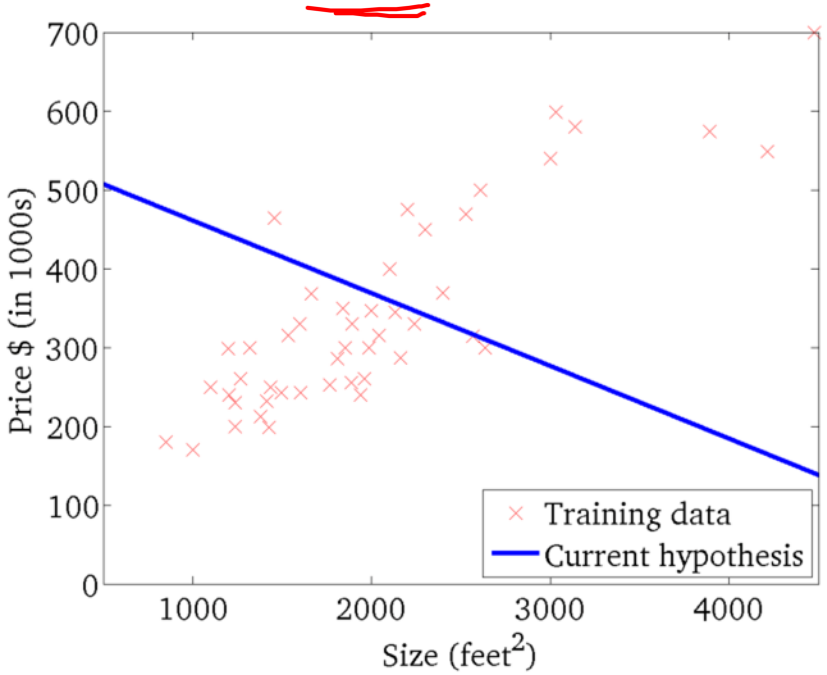


Source Credit : Slide by Andrew Ng

Gradient Descent

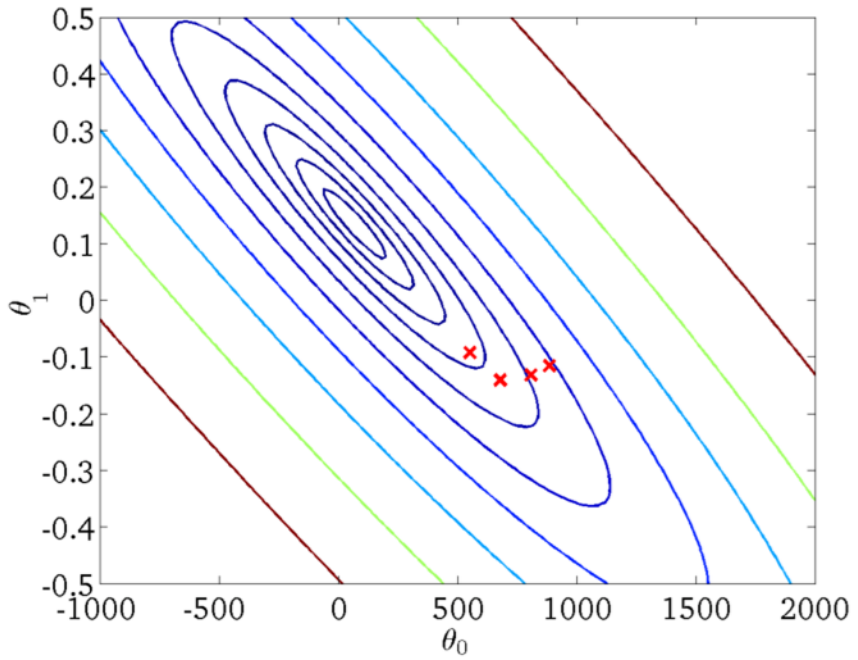
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

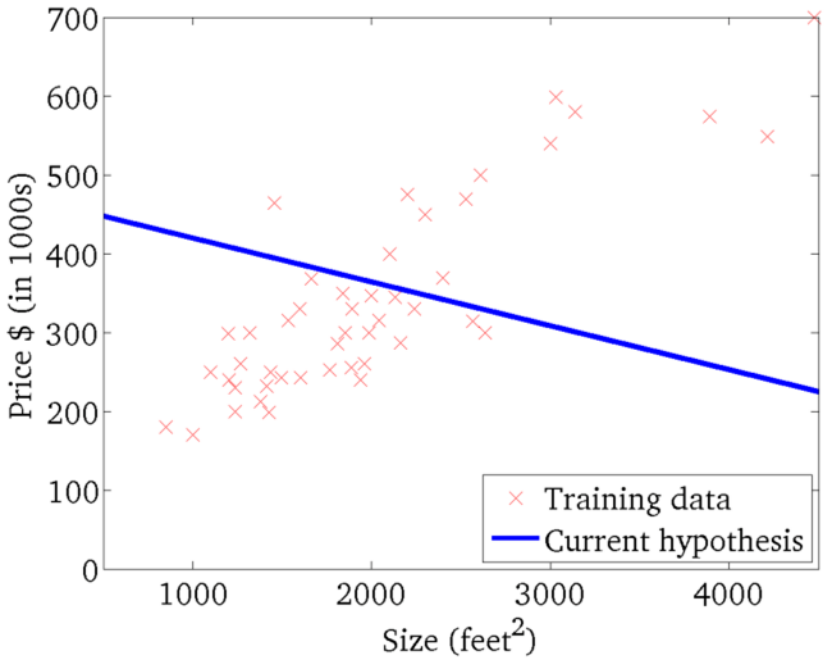


Source Credit : Slide by Andrew Ng

Gradient Descent

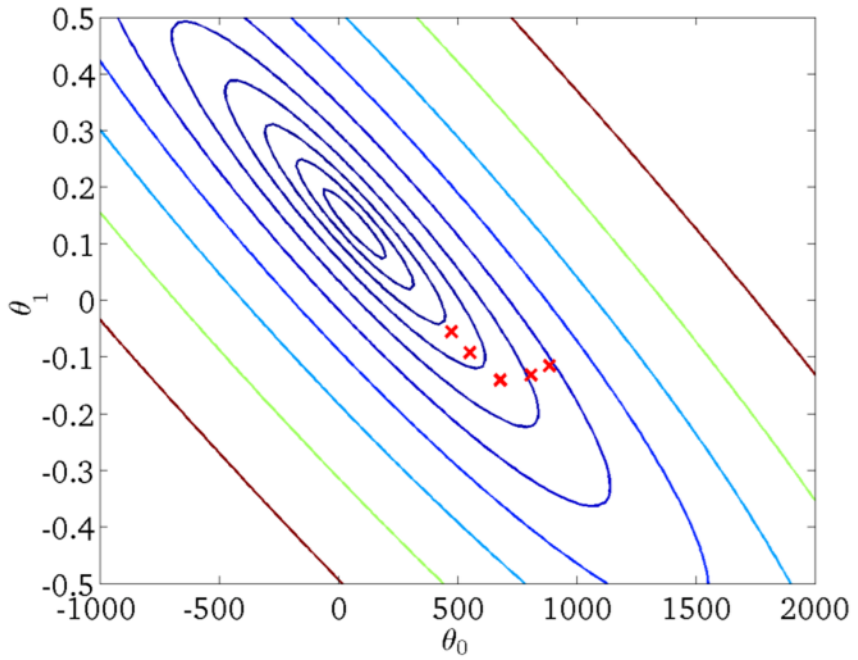
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

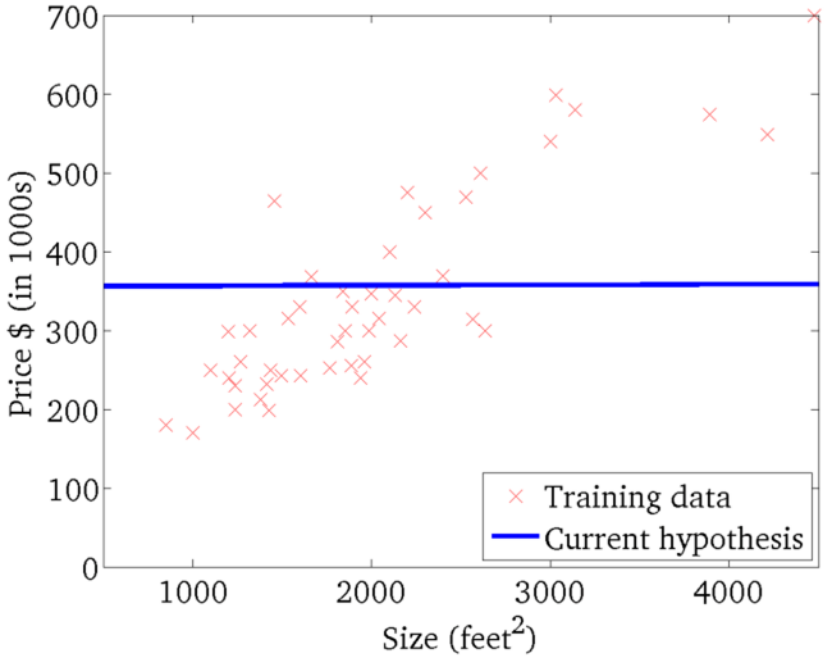


Source Credit : Slide by Andrew Ng

Gradient Descent

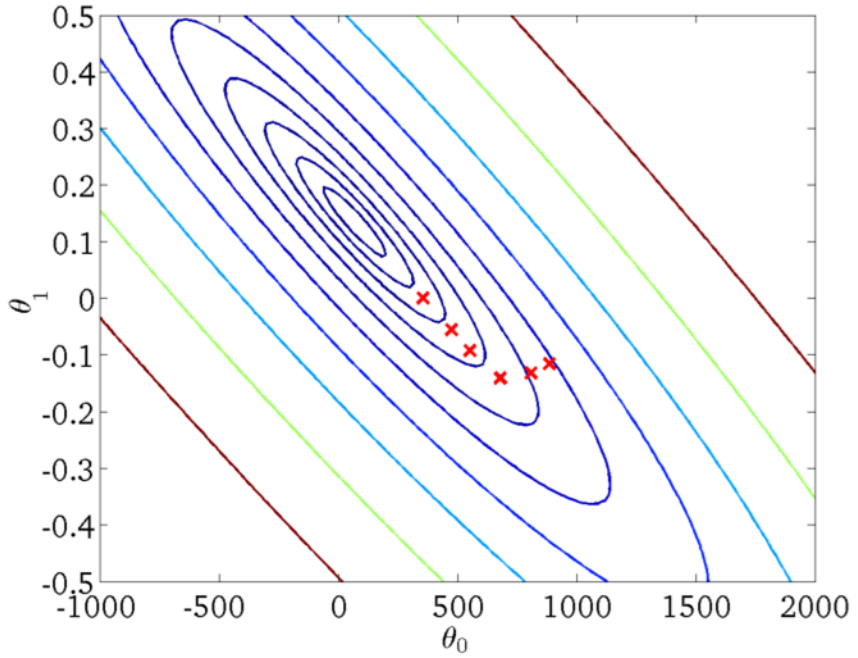
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

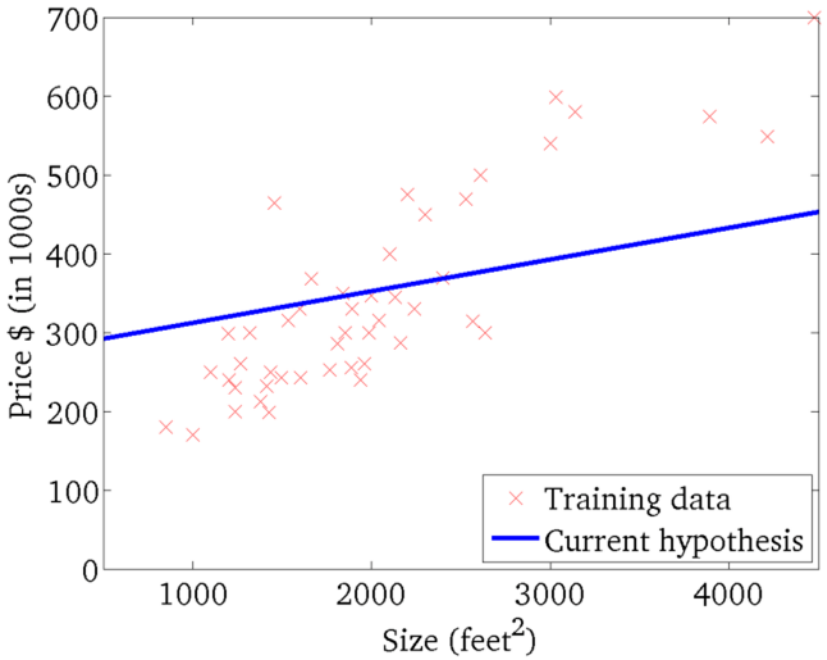


Source Credit : Slide by Andrew Ng

Gradient Descent

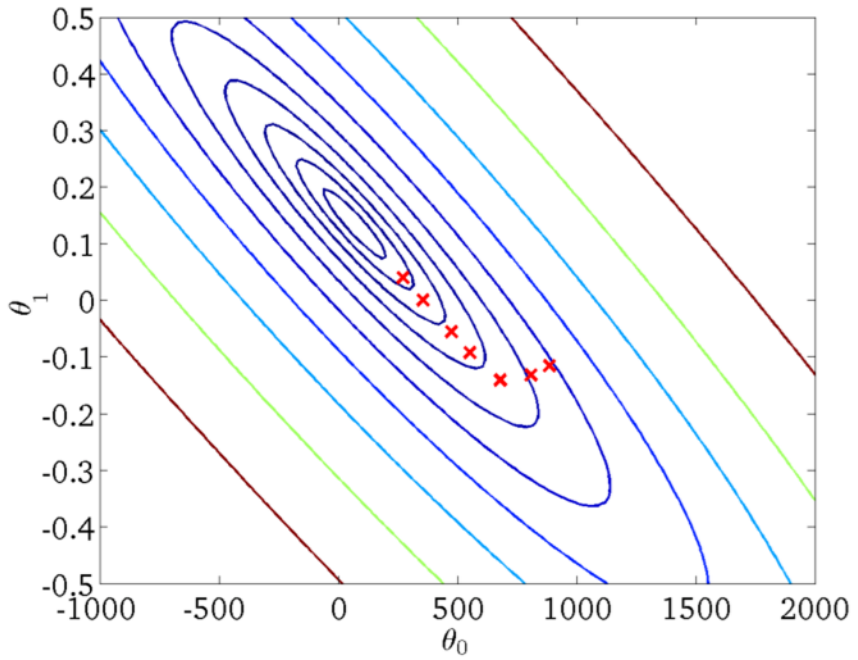
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

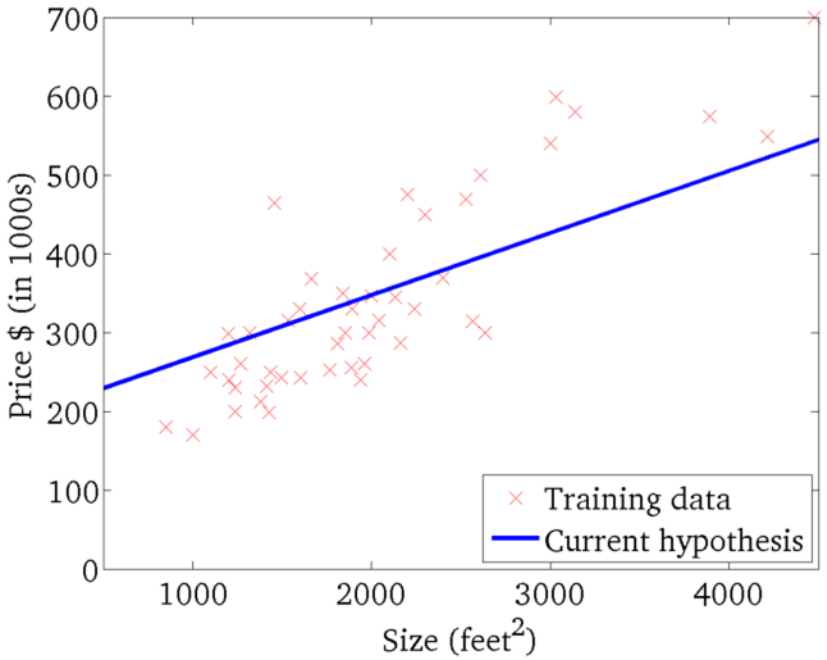


Source Credit : Slide by Andrew Ng

Gradient Descent

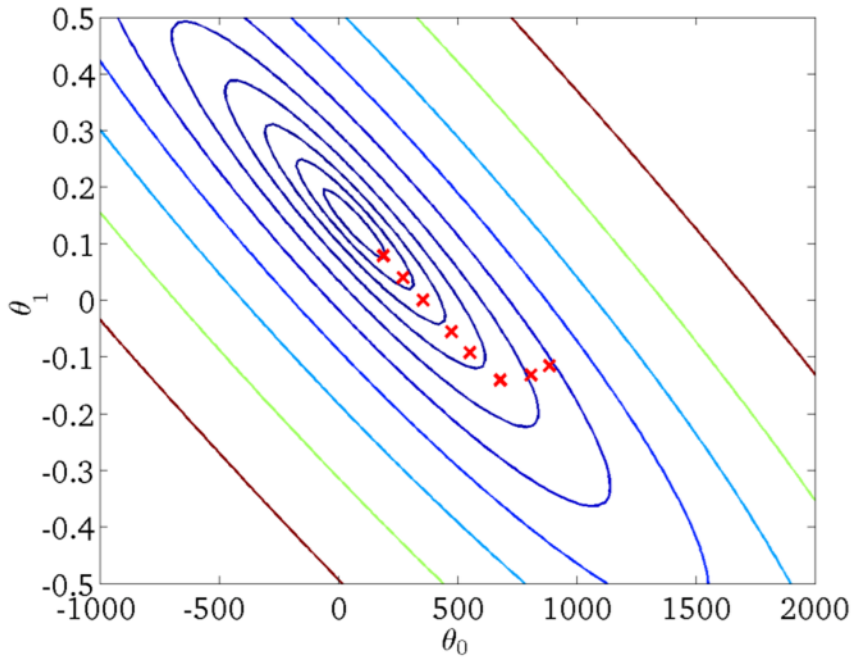
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

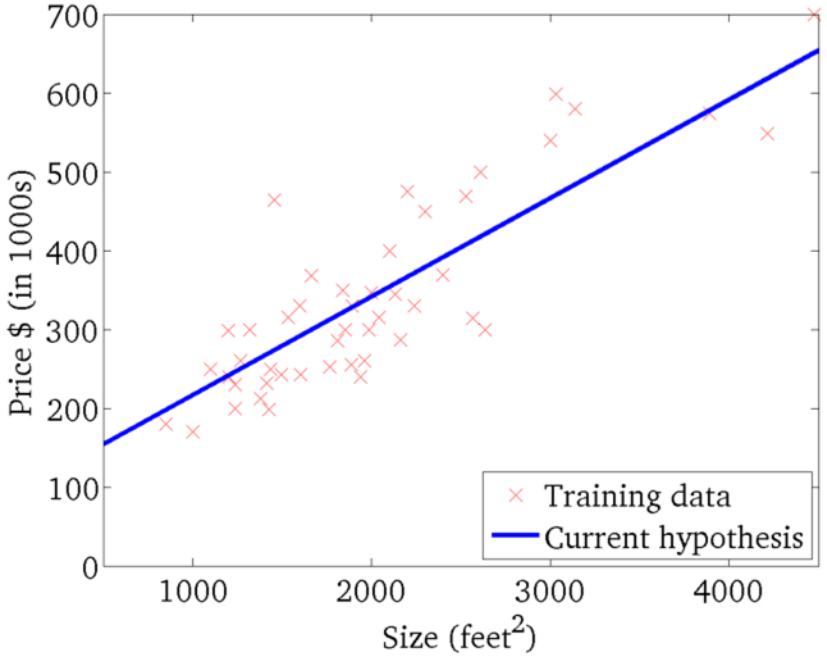


Source Credit : Slide by Andrew Ng

Gradient Descent

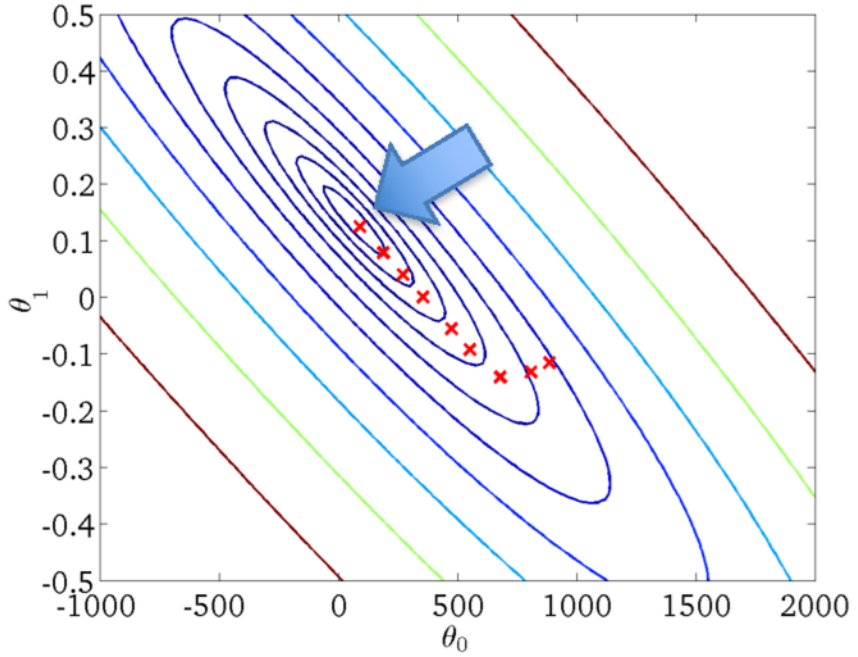
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



Source Credit : Slide by Andrew Ng

Gradient Descent

function should be convex

how to find??

minimize the cost function

$(h_\theta(x) - y_i)^2$

$h_\theta(x) = \theta_0 + \theta_1 x$

- Initialize θ → weight
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

new

simultaneous update
for j = 0 ... d

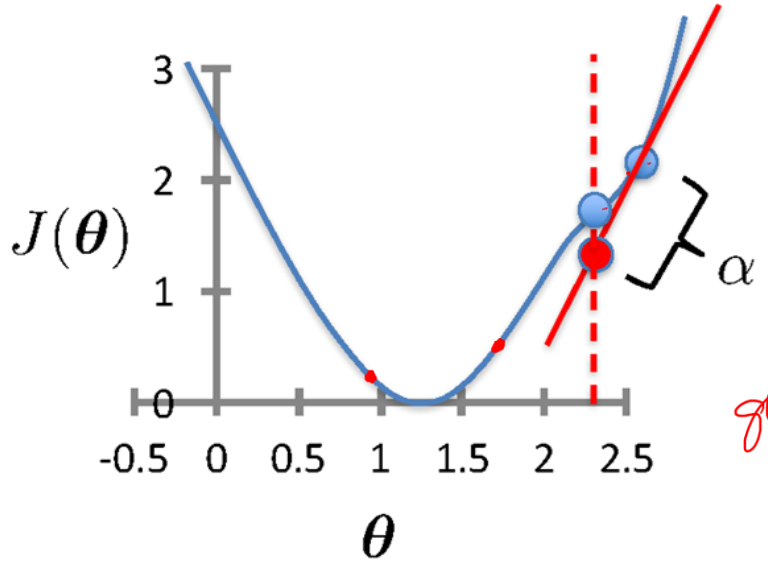
learning rate (small)
e.g., α = 0.05

step size

Hessian Matrix

$$\begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}$$

should be positive definite



global minimum

Gradient Descent

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

simultaneous update
for j = 0 ... d

For Linear Regression:

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)} \right)^2$$

$$= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right)^2$$

$$= \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right) \times \frac{\partial}{\partial \theta_j} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right)$$

$$= \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right) x_j^{(i)}$$

$\frac{d}{dx}(x^2) = 2x(1)$

$\frac{d}{da}(a-b)^2 = 2(a-b)\frac{d}{da}(a-b)$

$\frac{\partial}{\partial \theta_j}(y^{(i)}) = 0$

$\sum_{k=0}^{d} \theta_k x_k = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_j x_j + \cdots + \theta_d x_d$

substitute values for $k = 0, 1, 2 \ldots d$

$$\sum_{k=0}^{d} \theta_k x_k = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \underline{\theta_j^{\circ} x_j^{\circ}} + \ldots + \theta_d x_d$$

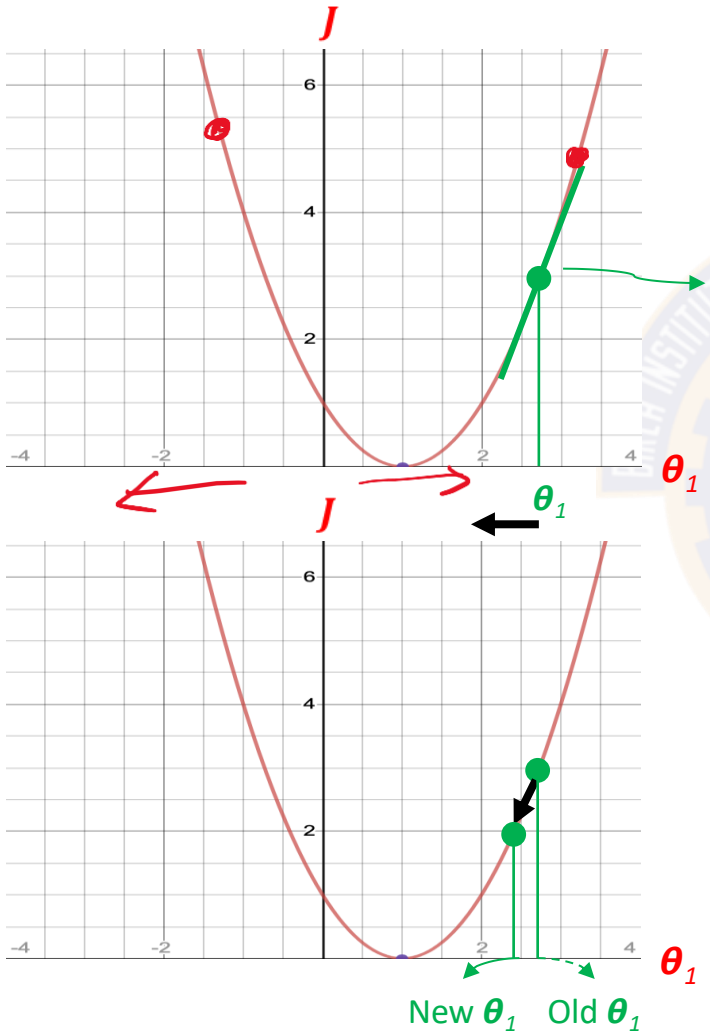
$$\frac{\partial}{\partial x}(2x) = 2$$

$$\frac{\partial}{\partial \theta_j^{\circ}}(\theta_j^{\circ} x_j^{\circ}) = x_j^{\circ} \checkmark$$

$$\frac{\partial}{\partial x_j^{\circ}}(\theta_j^{\circ} x_j^{\circ}) = \theta_j^{\circ}$$

$$\frac{\partial}{\partial \theta_j^{\circ}}(y^{i}) = 0 \text{ since } \underline{y^{i}} \text{ is independent of } \theta$$

Guarantee of Convergence



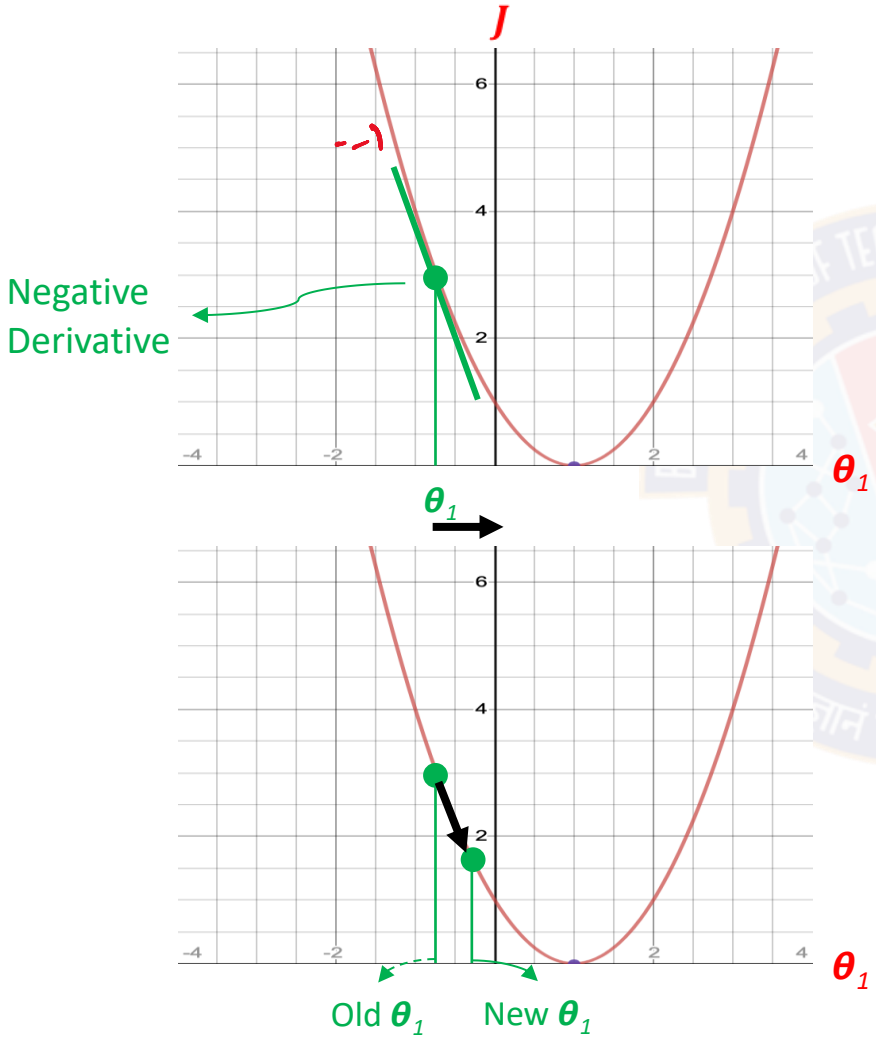
Source Credit: Prof. Mohammad Hammoud

$$\theta_1 = \theta_1 - \alpha \frac{d J(\theta_1)}{d \theta_j}$$
$$= \theta_1 - \alpha \, (Positive\ Number)$$

Decrease θ_1 by a certain value

$\alpha = 0.01$

Guarantee of Convergence



Source Credit: Prof. Mohammad Hammoud

$$\theta_1 = \theta_1 - \alpha \frac{d J(\theta_1)}{d \theta_j}$$

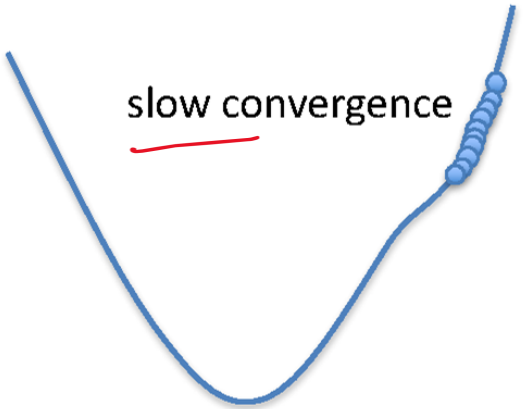
$$= \theta_1 - \alpha (Negative\ Number)$$

Increase θ_1 by a certain value

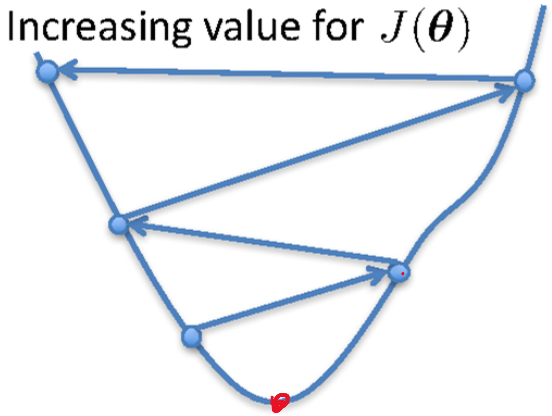
$\theta_1 - \alpha(-0.5) = \theta_1 + 0.5\alpha$

Choosing Learning Rate

α too small



α too large



- May overshoot the minimum
- May fail to converge
- May even diverge

To see if gradient descent is working, print out $J(\theta)$ each iteration

- The value should decrease at each iteration
- If it doesn't, adjust α

Gradient Descent algorithm : Effect of Feature Scaling

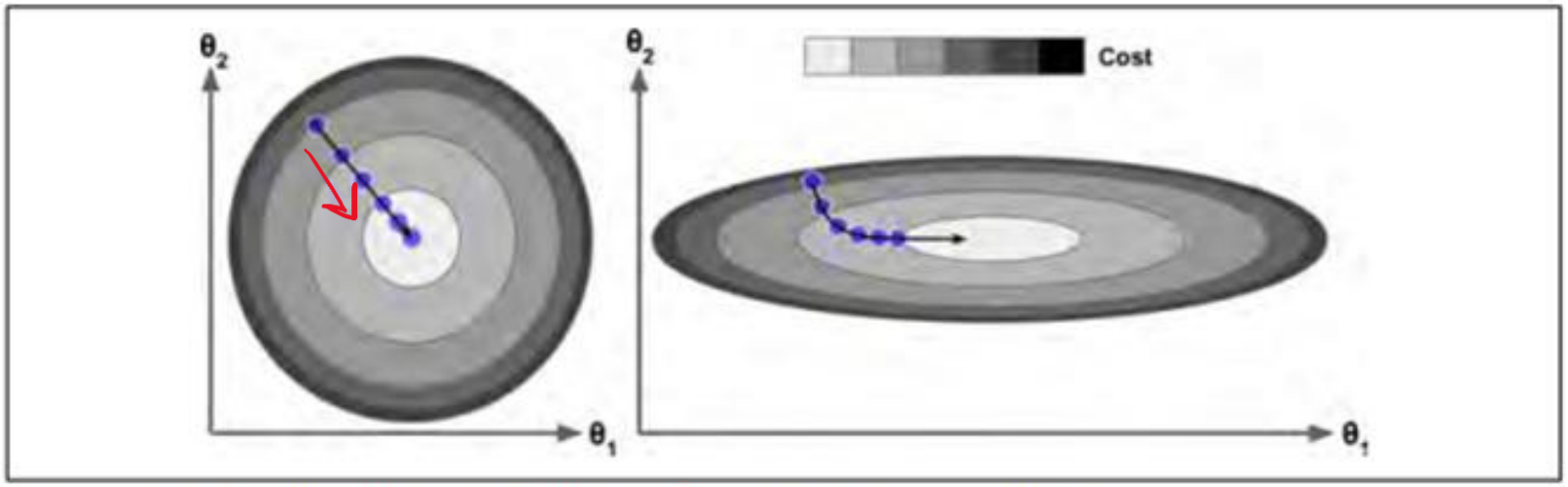


Figure 4-7. Gradient Descent with and without feature scaling

Aspect	With Scaling	Without Scaling
Contours shape	Circular	Elongated
Convergence speed	Fast	Slow
Gradient descent path	Straight to minimum	Zigzagging, inefficient

Why this happens?

Gradient descent uses the **slope** (partial derivatives) to update each parameter θ_j. When features have very different scales:

- One direction (e.g., θ_1) might change much faster than another (e.g., θ_2).
- This causes **inefficient updates**, as the algorithm overcorrects in one direction and undercorrects in another.
- The result is a **zigzagging path** instead of a direct descent.

Solution: **Feature Scaling**

Feature Scaling ensures:

- All features are on a **similar scale**, typically by:
  - **Standardization** (zero mean, unit variance)
  - **Min-max normalization** (rescale to [0, 1])
- This transforms the cost function contours into **circular shapes**.
- Gradient descent moves more **directly** toward the optimum.

Gradient Descent algorithm : Effect of Learning Rate

Problem Type 2 – Interpretation of Convergence or Effect of Hyper parameters

$\eta \rightarrow$ learning rate

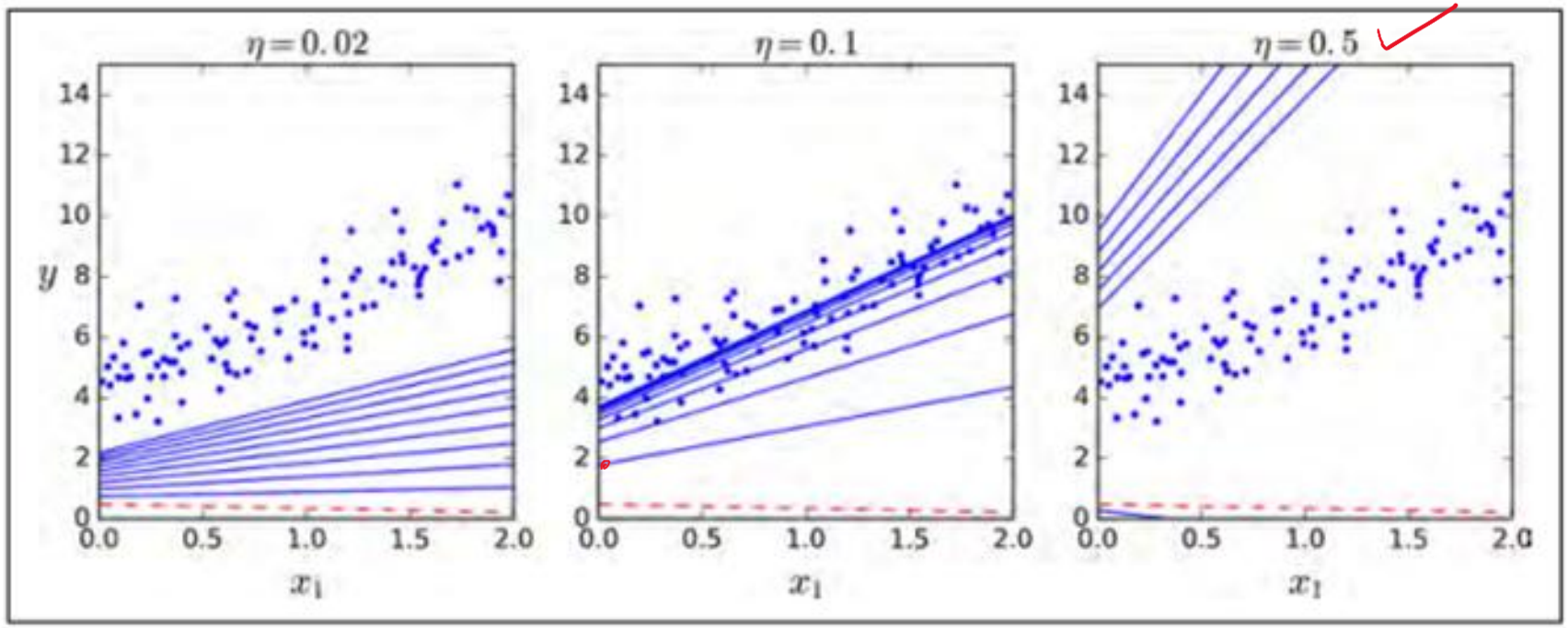


Figure 4-8. Gradient Descent with various learning rates

$J(\theta_2) = 1$

$J(\theta_1) = 1.1$

$J(\theta_2) = 1.23$

1
1.5
1.8

The **learning rate (η)** controls **how big the steps** are in gradient descent.

Learning Rate	Behavior	Result
Too Small	Slow, steady updates	Very slow convergence
Just Right	Efficient updates	Fast and stable convergence
Too Large	Overshooting, divergence	Unstable or no convergence

Gradient Descent for Linear Regression

- Initialize $\boldsymbol{\theta}$ weights
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{1}{n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right) x_j^{(i)}$$

simultaneous update for j = 0 ... d

- To achieve simultaneous update
  - At the start of each GD iteration, compute $h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right)$
  - Use this stored value in the update step loop
- Assume convergence when $\| \boldsymbol{\theta}_{new} - \boldsymbol{\theta}_{old} \|_2 < \epsilon$

L_2 norm: $$\| \boldsymbol{v} \|_2 = \sqrt{\sum_i v_i^2} = \sqrt{v_1^2 + v_2^2 + \ldots + v_{|v|}^2}$$

$v_1^0 = (v_1, v_2, v_3 \ldots v_n)$

|x|

$\|x\|_2$

L_2

Fit a linear Regression Line :

Gradient Descent : Problem Type 3

Steps :

(Assuming : 'n' no.of.instances and two predictors {x1, x2} and linear regression)

1. Identification of the equations $y = \theta_0 + \theta_1 x_1 + \theta_2 x_2$

2. Cost function & derivative

$$\theta_0' = \theta_0 - \frac{1}{n} * \text{learning rate} * (\text{sum } (\theta_0 + \theta_1 x_1 + \theta_2 x_2 - y))$$

$$\theta_1' = \theta_1 - \frac{1}{n} * \text{learning rate} * (\text{sum } (\theta_0 + \theta_1 x_1 + \theta_2 x_2 - y) * x_1)$$

$$\theta_2' = \theta_2 - \frac{1}{n} * \text{learning rate} * (\text{sum } (\theta_0 + \theta_1 x_1 + \theta_2 x_2 - y) * x_2)$$

3. Apply the equations

Fit a linear Regression Line :Gradient Descent

Fit a linear regression. Show only the first iteration of Gradient descent algorithm using learning rate of 0.02 for the following data , if the Relative Risk of Coronary Heart Disease is believed to be only linearly dependent on BMI as well as Diastolic Pressure. Assume the intercept of the regression model as 5 and the slope of independent variables as -0.03 (negative).

Patient	Systolic Pressure mm Hg	Diastolic Pressure mm Hg	BMI	Waist Circumference Threshold cm	RR-CHD (Relative Risk of Coronary Heart Disease)
1	140	80	35	100	1.81
2	120	80	25	80	1.22
3	130	100	30	60	1.71

1. Identification of the equations: RRCHD = 5 – 0.03 * BMI – 0.03 * Diastolic Pressure
2. Cost function & derivative
   1. $\theta_0' = \theta_0 - 1/3 * 0.02 * (\text{sum } (5\text{-}0.03*\text{BMI} - 0.03*\text{Diastolic Pressure} - \text{RRCHD}))$
      $= 5 - 1/3 * 0.02 * (\text{sum } (5\text{-}0.03*\text{BMI}\text{-}0.03*\text{Diastolic Pressure} - \text{RRCHD}))$
   2. $\theta_1' = \theta_1 - 1/3 * 0.02 * (\text{sum } (5\text{-}0.03*\text{BMI}\text{-}0.03*\text{Diastolic Pressure} - \text{RRCHD}) * \text{BMI})$
      $= -0.03 - 1/3 * 0.02 * (\text{sum } (5\text{-}0.03\text{BMI}\text{-}0.03\text{DiastolicPresure} - \text{RRCHD}) * \text{BMI})$
   3. $\theta_2' = \theta_2 - 1/3 * 0.02 * (\text{sum } (5\text{-}0.03*\text{BMI}\text{-}0.03*\text{Diastolic Pressure} - \text{RRCHD}) * \text{Diastolic Pressure})$
      $= -0.03 - 1/3 * 0.02 * (\text{sum } (5\text{-}0.03\text{BMI}\text{-}0.03\text{Diastolic Pressure} - \text{RRCHD}) * \text{Diastolic Pressure})$
3. Apply the equations : Answer at the end of first iteration:
   $\theta_0 = 5.0016$, $\theta_1 = 0.0476$, $\theta_2 = 0.179$

$\text{sum}(\theta_0 + \theta_1 x_1 + \theta_2 x_2) \rightarrow \hat{y}$

$$\theta_0' = \theta_0 - \frac{\alpha}{n}\left[\text{sum}(\theta_0 + \theta_1 x_1 + \theta_2 x_2) - y^i\right]$$

$$\theta_1' = \theta_1 - \frac{\alpha}{n}\left[\text{sum}(\theta_0 + \theta_1 x_1 + \theta_2 x_2) - y^i\right] \cdot x_1$$

$$\theta_2' = \theta_2 - \frac{\alpha}{n}\left[\text{sum}(\theta_0 + \theta_1 x_1 + \theta_2 x_2) - y^i\right] \cdot x_2$$

First iteration: For patient 1:

$$\hat{y} = 5 - 0.03(35) - 0.03(80) = 1.55$$

Actual $y = 1.81$, $\hat{y} = 1.55$

$$\text{Error}_1 = (\hat{y} - y) = 1.55 - 1.81 = -0.26$$

For patient 2:

$$\hat{y} = 5 - 0.03(25) - 0.03(80) = 1.85$$

$$\text{Error}_2 = 1.85 - 1.22 = 0.63$$

For patient 3: $\hat{y} = 5 - 0.03(30) - 0.03(100) = 1.1$

$$\text{Error}_3 = 1.1 - 1.71 = -0.61$$

Update the parameters:

$$\theta_0 = 5 - \frac{0.02}{3} \sum_{i=1}^{3} \text{Errors} = 5 - \frac{0.02}{3}[-0.26 + 0.63 - 0.61]$$

$$= 5.0016$$

Update θ_1 [BMI]:

$$\theta_1 = -0.03 - \frac{0.02}{3} \sum [\text{error} \cdot \text{BMI}]$$

$$\theta_1^{new} = -0.03 - \frac{0.02}{3}\left[(-0.26)(35) + (0.63)(25) - 0.61(30)\right]$$

$$= -0.03 + 0.07766 = 0.04766 \checkmark$$

Update θ_2 (Diastolic Pressure)

$$\theta_2 = -0.03 - \frac{0.02}{3}\left[\sum(\text{error} \cdot \text{diastolic pressure})\right]$$

$$= -0.03 + 0.20934 = 0.17934$$

$\varepsilon = 0.01$

$\varepsilon = 0.05$

Convergence criteria:

$$\| \theta_{new} - \theta_{old} \|_2 < \varepsilon$$

$a_1 x_1 + a_2 x_2 = b_1$

$a_3 x_1 + a_4 x_2 = b_2$

Closed Form Solution Vs. Gradient Descent

Gradient Descent	Closed Form Solution
• Requires multiple iterations • Need to choose α • Works well when n is large • Can support incremental learning	• Non-iterative • No need for α • Slow if n is large – Computing $(X^T X)^{-1}$ is roughly $O(n^3)$

Types of Gradient Descent Algorithms

Gradient Descent: Variants

- **Batch** gradient descent refers to calculating the derivative from all training data before calculating an update.
- **Minibatch** refers to calculating derivative of mini groups of training data before calculating an update.
- **Stochastic** gradient descent refers to calculating the derivative from each training data instance and calculating the update immediately

repeat until convergence {

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$$

$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

}

Update θ_0 and θ_1 simultaneously

$$temp0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$$

$$temp1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x^{(i)}$$

$$\theta_0 := temp0$$

$$\theta_1 := temp1$$

Gradient Descent: Variants

- **Batch** gradient descent refers to calculating the derivative from all training data before calculating an update.

Initialize the Parameters $(\theta_0^1, \theta_1^1, \ldots..)$
K=1
Repeat until Convergence {

$$\theta_0^{k+1} = \theta_0^k - \frac{\alpha}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$$

$$\theta_1^{k+1} = \theta_1^k - \frac{\alpha}{m} \sum_{i=1}^{m} ((h_\theta(x^{(i)}) - y^{(i)}) * x_1^{(i)})$$

$$\theta_2^{k+1} = \theta_2^k - \frac{\alpha}{m} \sum_{i=1}^{m} ((h_\theta(x^{(i)}) - y^{(i)}) * x_2^{(i)})$$

……

K=k+1
}
return $(\theta_0^1, \theta_1^1, \ldots..)$

Gradient Descent: Variants

- **Minibatch** refers to calculating derivative of mini groups of training data before calculating an update.

Divide the training instances into "N" batches each of size "m"
Initialize the Parameters $(\theta_0^1, \theta_1^1, \ldots..)$
K=1
Repeat until Convergence {
Repeat for every batch in 1 : N , each with 'm' instances {

$$\theta_0^{k+1} = \theta_0^k - \frac{\alpha}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$$

$$\theta_1^{k+1} = \theta_1^k - \frac{\alpha}{m} \sum_{i=1}^{m} ((h_\theta(x^{(i)}) - y^{(i)}) * x_1^{(i)})$$

$$\theta_2^{k+1} = \theta_2^k - \frac{\alpha}{m} \sum_{i=1}^{m} ((h_\theta(x^{(i)}) - y^{(i)}) * x_2^{(i)})$$

……
K=k+1
}
}
return $(\theta_0^1, \theta_1^1, \ldots..)$

150 data
15 groups
m=10

Gradient Descent: Variants

- **Stochastic gradient** descent refers to calculating the derivative from each training data instance and calculating the update immediately

Randomly shuffle training instances
Initialize the Parameters $(\theta_0^1, \theta_1^1, \ldots..)$
K=1
Repeat until Convergence {
Sample with replacement, only one random training instance "i" at a time

$$\theta_0^{k+1} = \theta_0^k - \alpha(h_\theta(x^{(i)}) - y^{(i)})$$
$$\theta_1^{k+1} = \theta_1^k - \alpha(h_\theta(x^{(i)}) - y^{(i)}) * x_1^{(i)}$$
$$\theta_2^{k+1} = \theta_2^k - \alpha(h_\theta(x^{(i)}) - y^{(i)}) * x_2^{(i)}$$

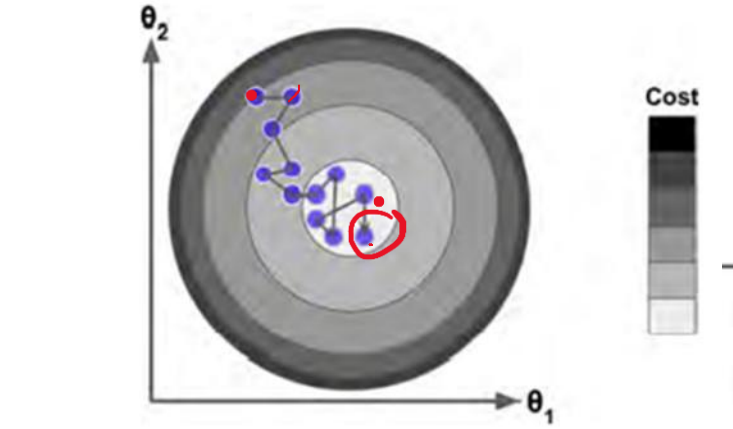
……
K=k+1
}
}
return $(\theta_0^1, \theta_1^1, \ldots..)$

Real-life Example:

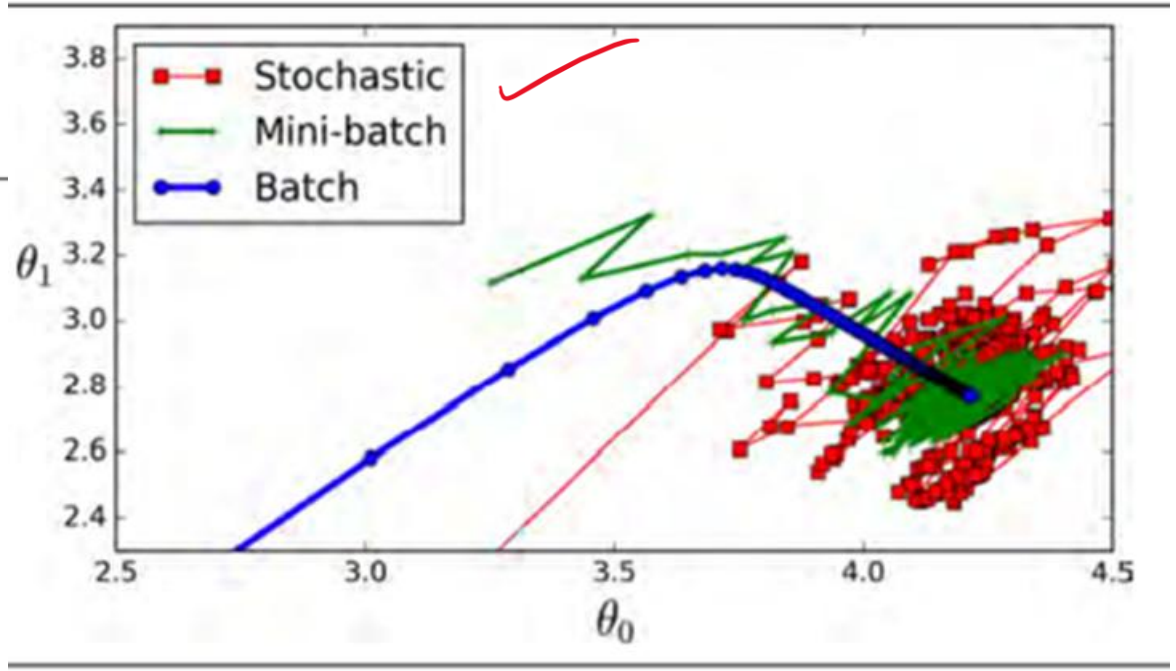
Think of three different types of hikers trying to reach the bottom of a foggy valley.

Method	The "Personality"	Their Philosophy
Batch	The **Perfectionist**	"I won't take a single step until I have calculated every possible detail perfectly."
Stochastic	The **Impulsive**	"I'm moving right now! Oh wait, maybe that way! No, this way!"
Mini-Batch	The **Pragmatist**	"I'll grab a handful of facts, make a solid move, and get the job done fast."

Gradient Descent: Variants - Effect on Convergence



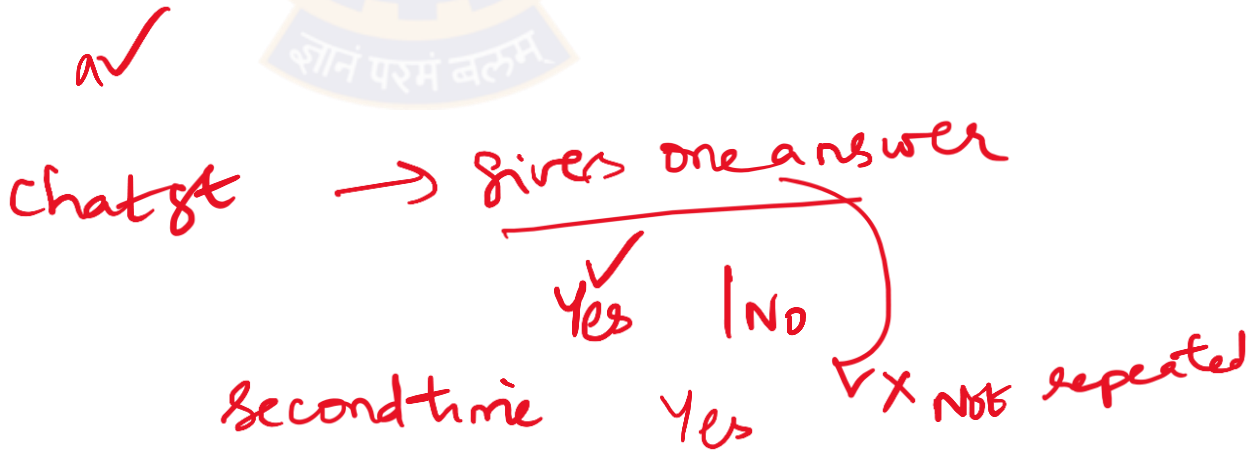
Stochastic Gradient Descent



1. Gradient Descent paths in parameter space

Epoch → complete passing through of all the datasets exactly once.

Type	Data Used per Update	Updates per Epoch	Speed	Stability	Common Use Case
Batch	All data	1	Slow	High	Small datasets
Stochastic	1 sample	n (number of samples)	Fast	Low	Online learning
Mini-Batch	Small batches	n / batch_size	Medium	Medium	Deep learning (default)



Evaluation Metrics

Evaluation of Linear Regression Model

Mileage (in kmpl)	*Car Price (in cr)*
9.8	10.48
9.12	1.75
9.5	6.95
10	2.51
….	…..

Mean Square Error

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$

Mean Absolute Error

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}|$$

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}$$

coefficient of determination

$$R^2 = 1 - \frac{SS_{residual}}{SS_{Total}}$$

Unseen Data	
Mileage (in kmpl)	***Car Price (in cr)***
7.5	9.25
10	6.5
….	…..

Model 1

CarPrice = 8.5 + 0.5 Mileage – 1.5 Mileage^2

Model 2

CarPrice = -5.5 + 1.5 Mileage

Evaluation of Linear Regression Model

R-squared

Mileage (in kmpl)	*Car Price (in cr)*
9.8	10.48
9.12	1.75
9.5	6.95
10	2.51
….	…..

variation in 'y' that is explained by a regression model

$$explained\ variation = \hat{y} - \overline{y}$$

variation in 'y' that is not captured/explained by a regression model

$$unexplained\ variation = y - \hat{y}$$

$$total\ variation = (y - \hat{y}) + (\hat{y} - \overline{y}) = (y - \overline{y})$$

$$R^2 = 1 - \frac{SS_{residual}}{SS_{Total}}$$

$$SS_{explained} = \sum_{i=1}^{n} (\hat{y}_i - \overline{y})^2$$

$$SS_{residual} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$SS_{Total} = \sum_{i=1}^{n} (y_i - \overline{y})^2$$

where :

$SS_{explained}$ = *explained variation sum of squares*

$SS_{residual}$ = *unexplained variation sum of squares*

SS_{Total} = *total variation sum of squares*

Model 1

$$\hat{y}\ CarPrice = 8.5 + 0.5\ Mileage - 1.5\ Mileage^2$$

x

- **variation that is explained by a regression model**
- **measures the goodness of fit of a regression model**

$\hat{y}$ → predicted y-value

Car Price (in cr)
10.48
1.75
6.95
2.51
Mean Y

$y = ax + b$

$a_0 x_0 + a_1 x_1 + a_2 x_2^2 + a_3 x_3^3 + \ldots + a_n x_n^n$

Linear Basis Models

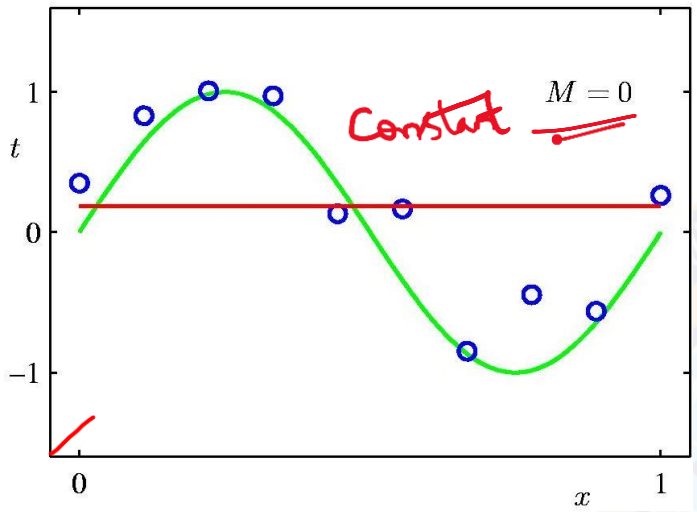
x, $y = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$

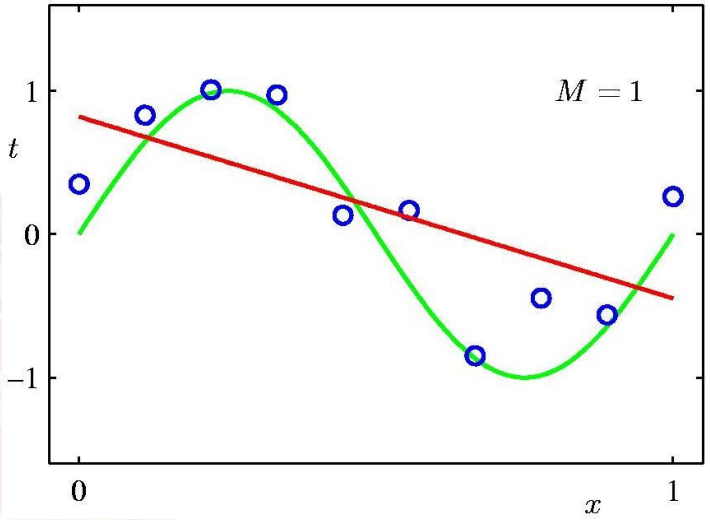
What if output is a non-linear function of input vector?

Polynomial Regression

degree of the polynomial

$y = ax^M + b$



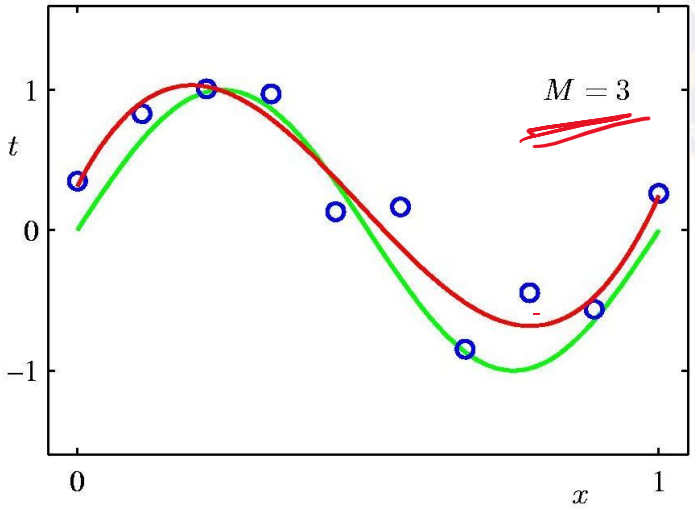


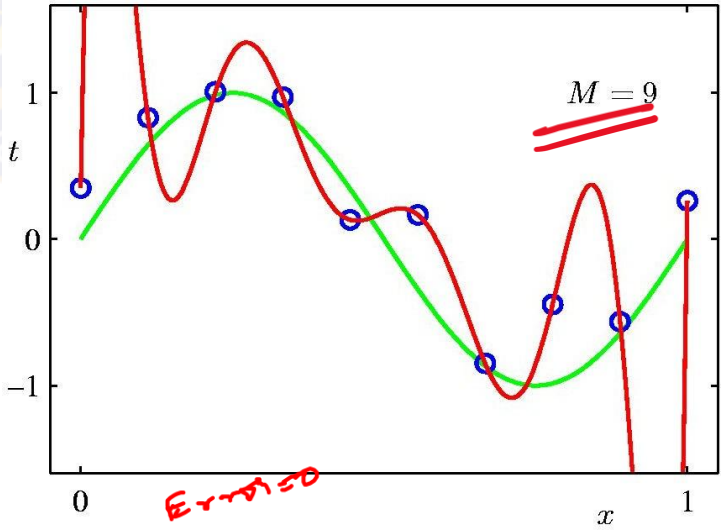
When M = 1

$y = ax + b$

when m = 2

$y = ax^2 + b$





when m = 0

$y = k$

overfitting

Linear Basis Function Models

- The inputs **X** for linear regression can be:
  - Original quantitative inputs
  - Transformation of quantitative inputs
    - e.g. log, exp, square root, square, etc.
  - Polynomial transformation
    - example: $y = \beta_0 + \beta_1 \cdot x + \beta_2 \cdot x^2 + \beta_3 \cdot x^3$
  - Basis expansions
  - Dummy coding of categorical inputs
  - Interactions between variables
    - example: $x_3 = x_1 \cdot x_2$

This allows use of linear regression techniques to fit non-linear datasets.

Input X	Output Y
exp(2)	....
exp(4)	
exp(6.3)	
exp(9.2)	

X No.of.Years of Experience (in Years)	X^2	Y Salary Of the Employee (in Lakhs)
1	1	2
2	4	3
3	9	4
4	16	5
5	25	6

X1 = Graduate	X2 = PostGraduate	X3 = Others	Y Salary Of the Employee
0	0	1	2
1	0	0	3
0	0	1	4
0	1	0	5
1	0	0	6

Linear Basis Function Models

Example: an M-th order polynomial function of one dimensional feature x :

$$y(x, \mathbf{w}) = w_0 + \sum_{j=1}^{M} w_j x^j$$

where x^j = j-th power of x

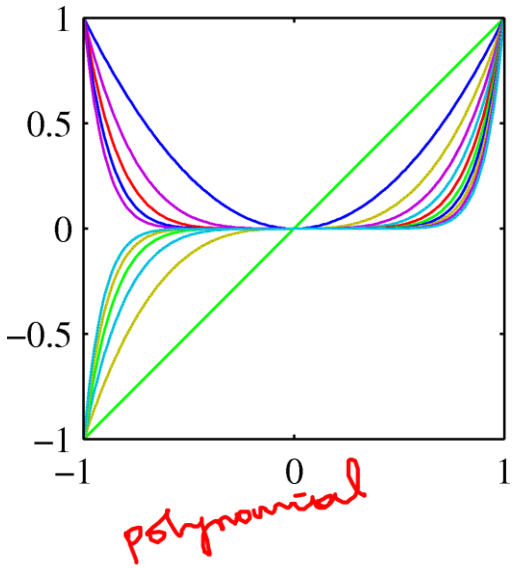
$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

X No.of.Years of Experience (in Years)	X^2	Y Salary Of the Employee (in Lakhs)
1	1	2
2	4	3
3	9	4
4	16	5
5	25	6

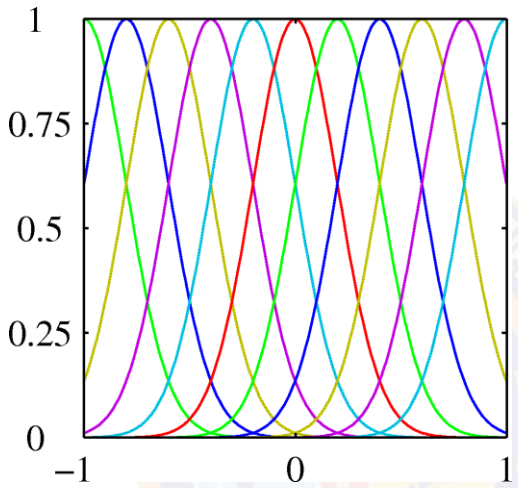
$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x})$$

- $\Phi_j(x)$ are known as *basis functions*. Typically, $\Phi_0(x) = 1$, so that w_0 acts as a bias.
- In the simplest case, we use linear basis functions : $\Phi_d(x) = x^d$.
- They are called linear models because this function is linear in **w**.

Linear Basis Function Models - Examples



polynomial

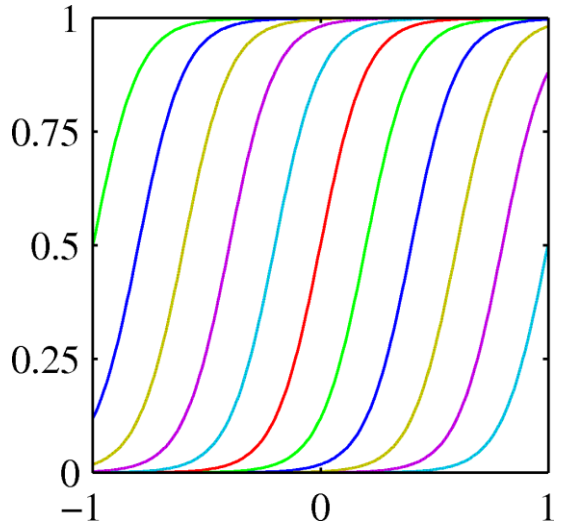


Normal distribution

Gaussian basis functions:

$$\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$$

These are local; a small change in x only affect nearby basis functions. μ_j and s control location and scale (width).



Sigmoidal basis functions:

where $\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right)$

$$\sigma(a) = \frac{1}{1+\exp(-a)}.$$

Also these are local; a small change in x only affect nearby basis functions. μ_j and s control location and scale (slope).

Notion of Bias - Variance

Bias and Variance

1. What is Bias?

- **Bias** = Error from wrong assumptions in the model.
- It tells how far the model's predictions are from the *true relationship*.

High Bias means the model is **too simple** (it underfits).

2. What is Variance?

- **Variance** = Error from sensitivity to training data.
- It tells how much the model's predictions would change if we trained on a different dataset.

High Variance means the model is **too complex** (it overfits).

- Example: A very wiggly polynomial that perfectly fits training points but fails on new data.

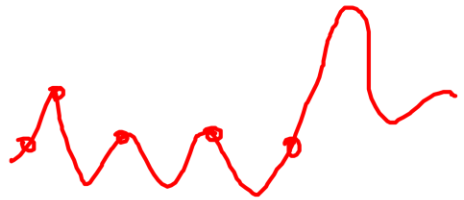
Underfitting and Overfitting

Underfitting

- Happens when the model is **too simple** to capture the patterns in the data.
- It has **high bias** (wrong assumptions) and usually **low variance** (stable but inaccurate).

Symptoms:

- Both **training error** and **test error** are high.
- Model cannot learn even training data well.



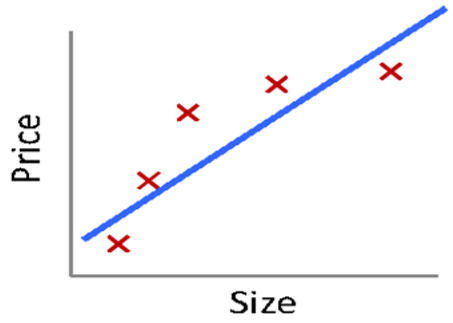
Overfitting

- Happens when the model is **too complex** and tries to learn not only the true pattern but also the **noise** in the training data.
- It has **low bias** (fits training data very well) but **high variance** (fails on new/unseen data).

Symptoms:

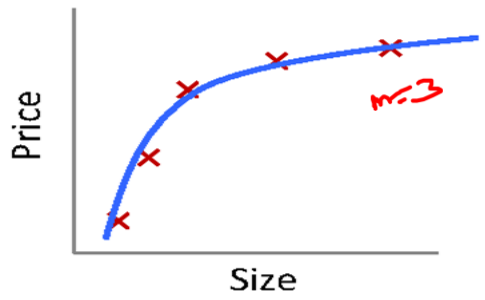
- **Training error is very low**.
- **Test error is very high** (poor generalization).

Quality of Fit



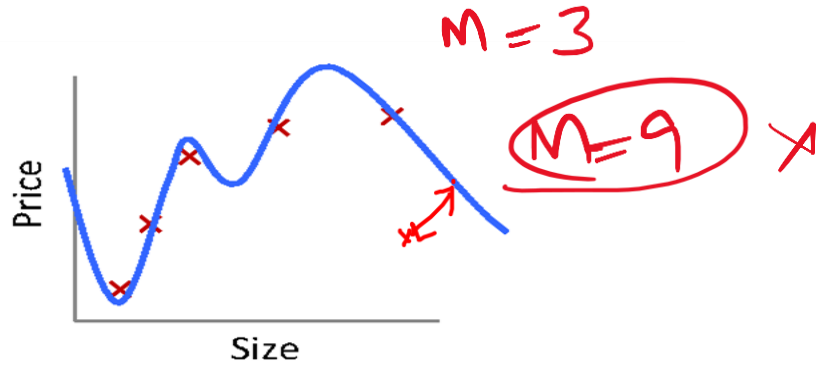
$\theta_0 + \theta_1 x$

Underfitting
(high bias)



$\theta_0 + \theta_1 x + \theta_2 x^2$

Correct fit



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

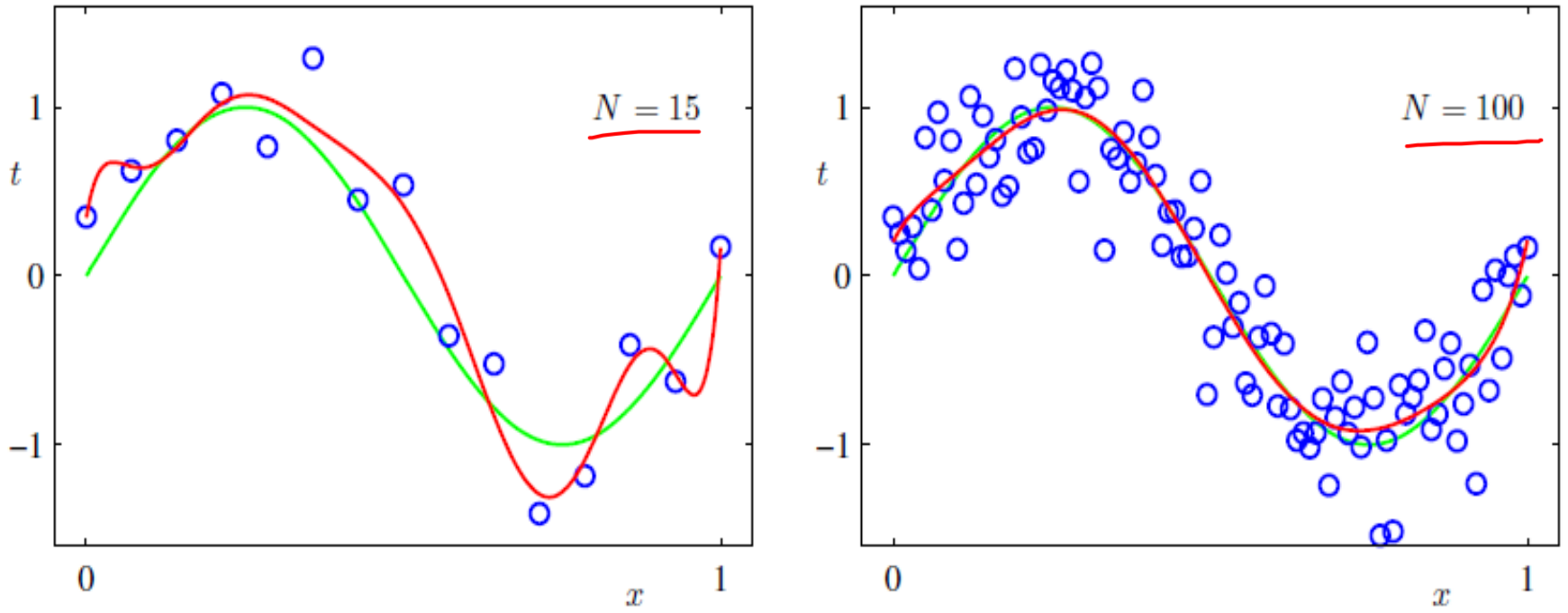
Overfitting
(high variance)

Overfitting:

- The learned hypothesis may fit the training set very well ($J(\boldsymbol{\theta}) \approx 0$)
- ...but fails to generalize to new examples

Handling Overfitting – Way 1

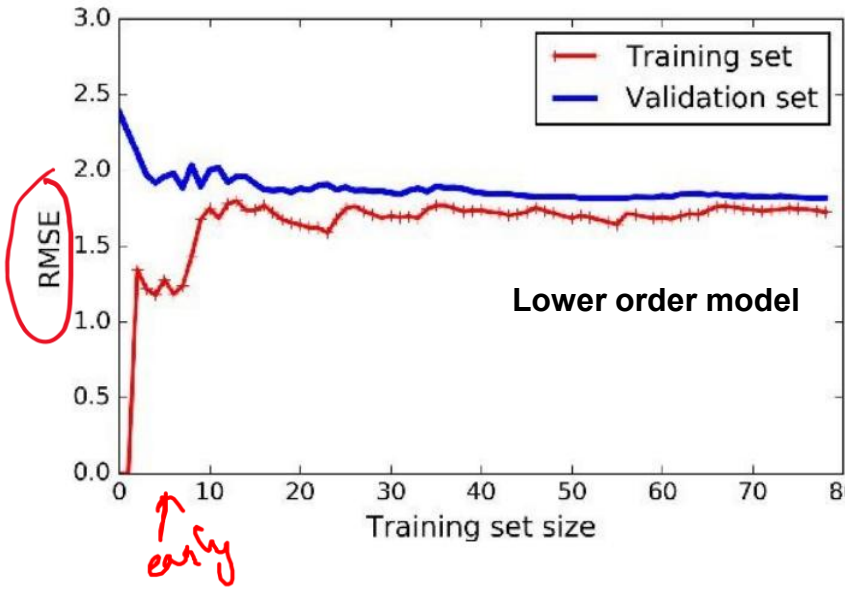
- Increase in Size of the data set reduces the over-fitting problem

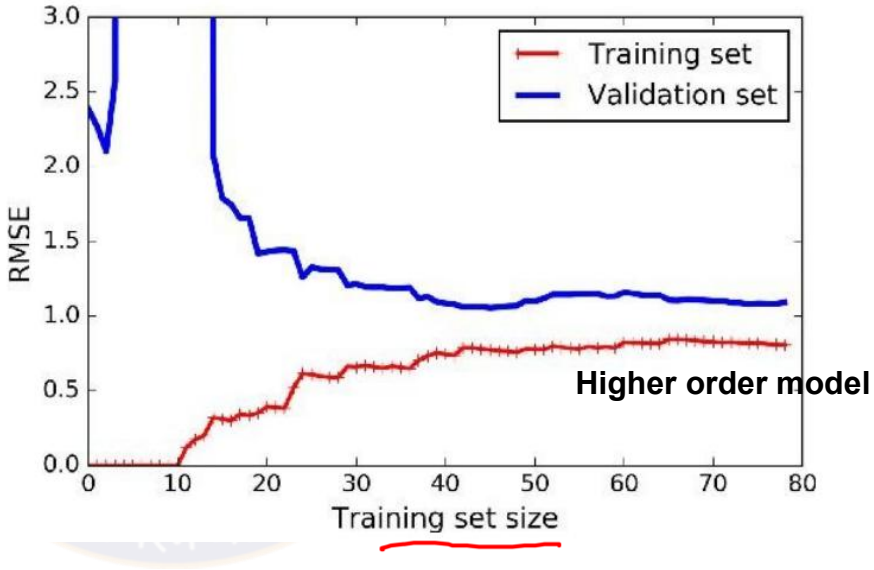


Effect of Training Size on Over fitting

Problem Type 4 : Interpretation of the Model Fit

- Size of training dataset needs to be large to prevent when higher order model is used.





This model is suffering from high bias.
It is underfitting the data

- This model has **high variance**
- It is **overfitting**

Polynomial Fitting can lead to Overfitting

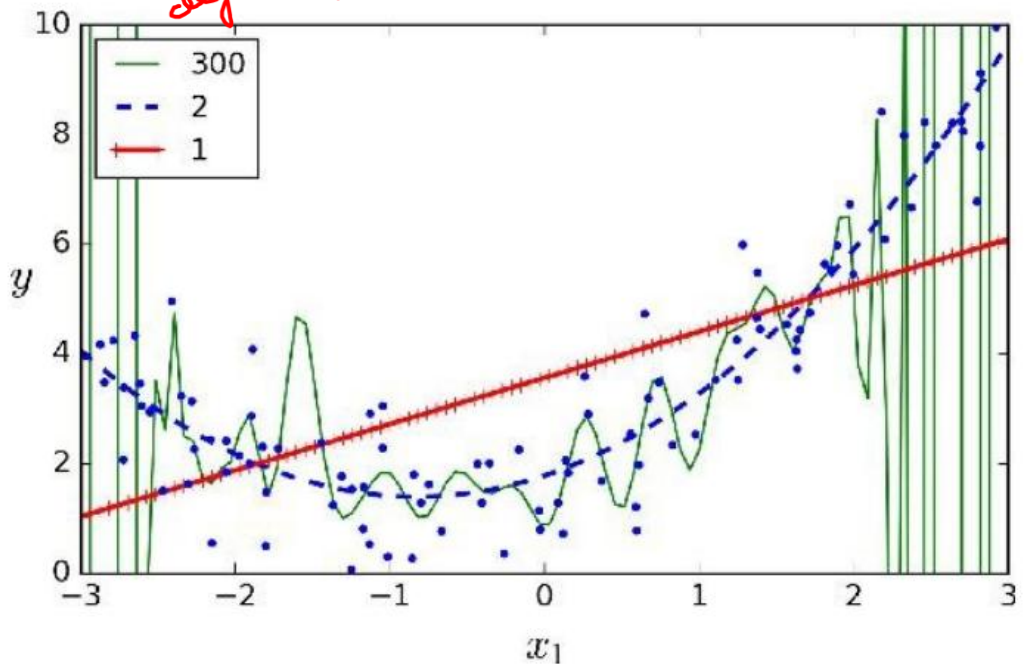
Handling Overfitting – Way 2 – Reduce the complexity of the model

Underlying target function is quadratic

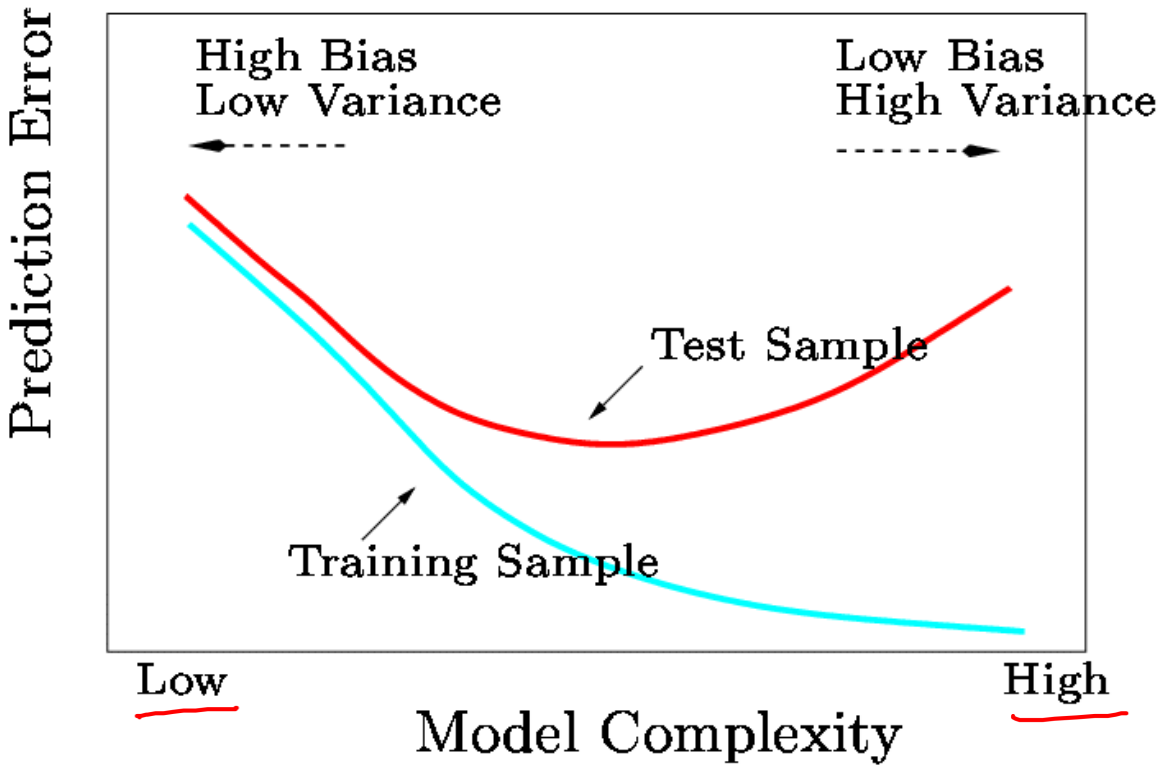
features scaling ✓

- Linear model results in under fitting with large bias
- Polynomial of order 300 results in a large variance

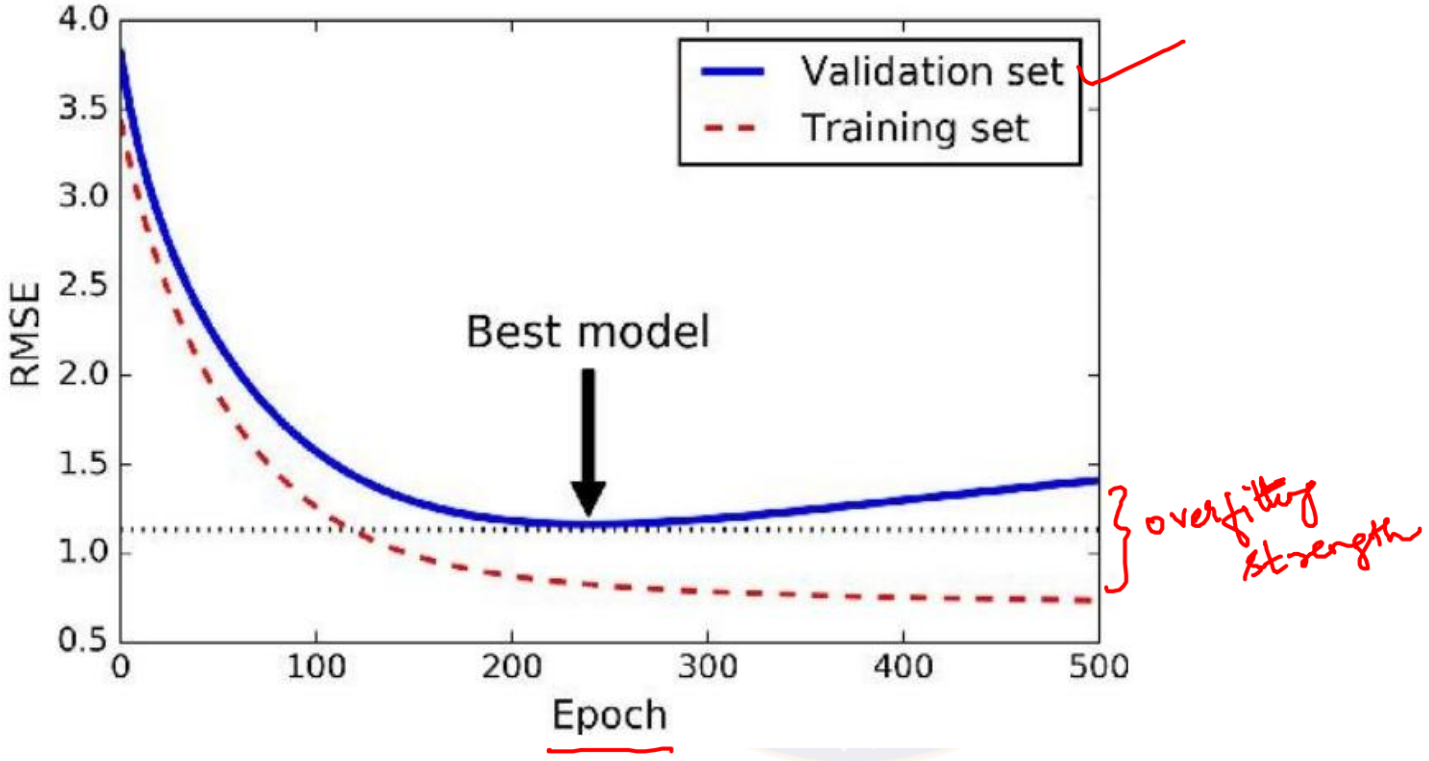
degree of polynomial



How to experiment on Model Complexity ?



Handling Overfitting – Way 3



overfitting strength

- **X-axis (Epoch):** Number of training iterations.
- **Y-axis (RMSE):** Root Mean Square Error — lower values mean better predictions.

We select the model at the epoch where **validation RMSE is lowest**, not where training error is lowest.

Regularization

(*This notion is common for both Linear Regression – Module 3 and Logistic Regression – Module 4)

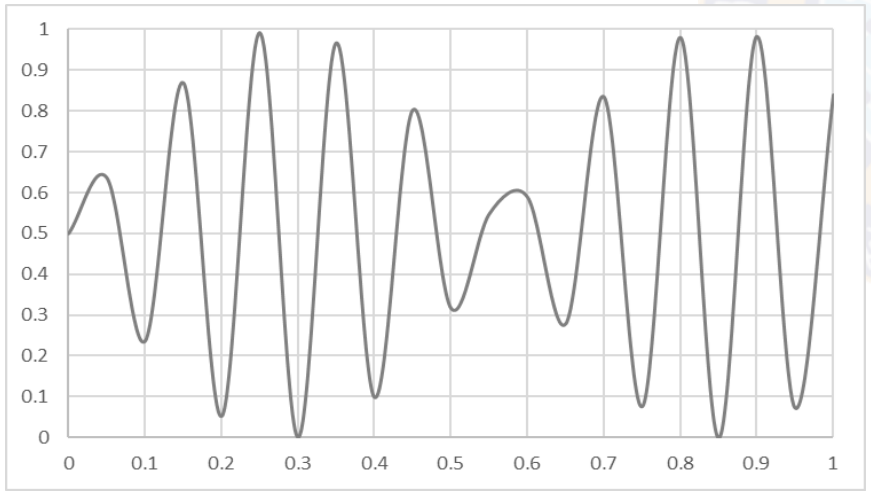
Overfitting vs Underfitting

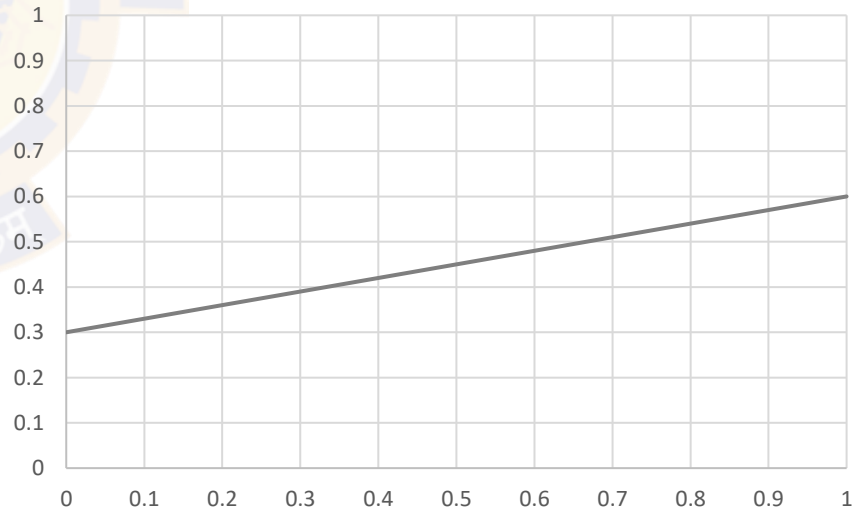
Overfitting

- Fitting the data too well
  - Features are noisy / uncorrelated to concept

Underfitting

- Learning too little of the true concept
  - Features don't capture concept
  - Too much bias in model





Regularization

- A method for automatically controlling the complexity of the learned hypothesis
- **Idea**: penalize for large values of θ_j
  - Can incorporate into the cost function
  - Works well when we have a lot of features, each that contributes a bit to predicting the label
- Can also address overfitting by eliminating features (either manually or via model selection)

Regularization

$(\hat{y} - \bar{y})^2$

- Linear regression objective function

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{d} \theta_j^2$$

$\theta_j \rightarrow$ weights

model fit to data — regularization

  - λ is the regularization parameter ($\lambda \geq 0$)
  - No regularization on θ_0!

bias

Understanding Regularization

$$J(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{d}\theta_j^2$$

- Note that $\sum_{j=1}^{d}\theta_j^2 = \|\boldsymbol{\theta}_{1:d}\|_2^2 = \theta_1^2 + \theta_2^2 + \theta_3^2 + \ldots + \theta_d^2$
  - This is the magnitude of the feature coefficient vector!

- We can also think of this as:

$$\sum_{j=1}^{d}(\theta_j - 0)^2 = \|\boldsymbol{\theta}_{1:d} - \vec{\mathbf{0}}\|_2^2$$

L_2 norm

  - L_2 regularization pulls coefficients toward 0

Norm

A norm is a mathematical function that measures the size or length of a vector.

For a vector

$$\theta = [\theta_1, \theta_2, \ldots, \theta_d],$$

the **L2 norm** is defined as:

$$\|\theta\|_2 = \sqrt{\theta_1^2 + \theta_2^2 + \cdots + \theta_d^2}.$$

This is the standard Euclidean distance from the origin to the point $(\theta_1, \theta_2, \ldots, \theta_d)$.

Often in machine learning, we use the **squared L2 norm**:

$$\|\theta\|_2^2 = \theta_1^2 + \theta_2^2 + \cdots + \theta_d^2,$$

because it is easier to work with mathematically (no square root).

Another commonly used norm is the **L1 norm**:

$$\|\theta\|_1 = |\theta_1| + |\theta_2| + \cdots + |\theta_d|.$$

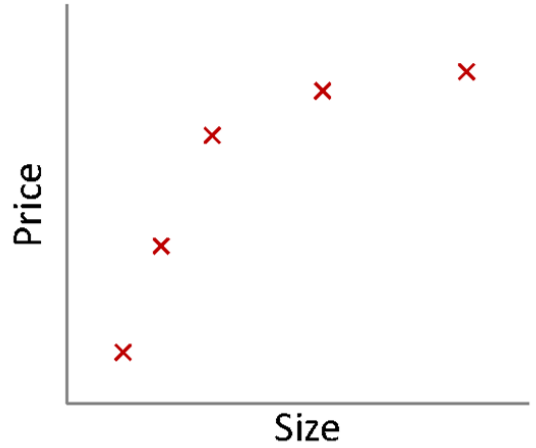
This measures distance by summing the absolute values of the components. It is sometimes called the Manhattan norm, because it is like measuring distance along grid lines instead of diagonally.

Understanding Regularization

$$J(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{d}\theta_j^2$$

- What happens if we set λ to be huge (e.g., 10^{10})?

$\lambda \gg 0$



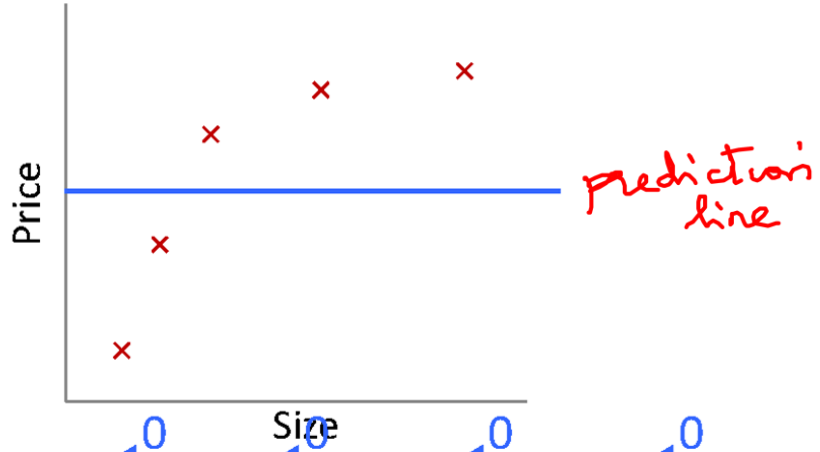
$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Based on example by Andrew Ng

Understanding Regularization

$$J(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{d}\theta_j^2$$

- What happens if we set λ to be huge (e.g., 10^{10})?



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

0 0 0 0

Based on example by Andrew Ng

Interpretation:

When λ is huge

- The penalty term dominates the cost function.
- To minimize the cost, the model forces:
- $\theta_1 \approx 0,\ \theta_2 \approx 0,\ \theta_3 \approx 0,\ \theta_4 \approx 0$
- Only θ_0 (bias term) remains significant.

$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_4 x_4$

The model becomes:

$$y \approx \theta_0$$

That means:

- The prediction is a **constant horizontal line** (shown in blue).
- It ignores the relationship between input (size) and output (price).
- It simply predicts the average value of the target.

This situation leads to **underfitting**:

- Model is too simple.
- Cannot capture patterns in data.
- High training error and high validation error.

Regularization controls model complexity:

- Small $\lambda \rightarrow$ complex model $\rightarrow$ risk of overfitting
- Moderate $\lambda \rightarrow$ balanced model $\rightarrow$ good generalization
- Very large $\lambda \rightarrow$ overly simple model $\rightarrow$ underfitting

Regularization Ridge Regression / Tikhonov regularization

- Cost Function

$$J(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{d}\theta_j^2$$

regularisation

- Fit by solving $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

- Gradient update:

$\frac{\partial}{\partial\theta_0}J(\theta)$

$$\theta_0 \leftarrow \theta_0 - \alpha\frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)$$

$\frac{\partial}{\partial\theta_j}J(\theta)$

$$\theta_j \leftarrow \theta_j - \alpha\frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)x_j^{(i)} - \lambda\theta_j$$

regularization

constant

- We can rewrite the gradient step as:

$$\theta_j \leftarrow \theta_j\,(1-\alpha\lambda) - \alpha\frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$$

Regularization

Lasso Regression (Least Absolute Shrinkage and Selection Operator Regression)

L_1 Regularisation

- Cost Function

$$J(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{d}|\theta_j|$$

- Fit by solving $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

- Gradient update:

$\frac{\partial}{\partial\theta_0}J(\theta)$

$$\theta_0 \leftarrow \theta_0 - \alpha\frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)$$

$\frac{\partial}{\partial\theta_j}J(\theta)$

$$\theta_j = \theta_j - \frac{\alpha}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)\boldsymbol{x}_j^{(i)} - \alpha\lambda\,\text{sign}(\theta_j)$$

regularization

where $\text{sign}(\theta_i) = \begin{cases} -1 & \text{if } \theta_i < 0 \\ 0 & \text{if } \theta_i = 0 \\ +1 & \text{if } \theta_i > 0 \end{cases}$

Regularization Elastic Net

- Cost Function

$$J(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2 + r\frac{\lambda}{2}\sum_{j=1}^{d}\theta_j^2 + \frac{(1-r)}{2}\lambda\sum_{j=1}^{d}|\theta_j|$$

Combination of Lasso regularisation & Ridge regression

Control the regularization using **the Mix Ration "r"**:
When,
r = 0, Elastic Net is equivalent to Ridge Regression,
r = 1, it is equivalent to Lasso Regression

from sklearn.linear_model import ElasticNet
ElasticNet(alpha=0.1, l1_ratio=0.5)

How to choose the right Regularization? Common Usage & Observation

- L1 regularization has the ability to set some coefficients to 0 exactly leading to a *sparse* model
- L1 regularization helps in feature selection by eliminating the features that are not important
- **L1 cannot be used efficiently in gradient-based approaches since it is not-differentiable unlike L2**
- L2 will in general lead to small magnitudes of weights but not exactly 0.
- Elastic net – Prefer in Highly correlated features

Numerical Exercise – For Student Practice

Fit a linear regression. Show only the first iteration of Gradient descent algorithm using learning rate of **0.02** for the following data , if the Relative Risk of Coronary Heart Disease is believed to be only linearly dependent on BMI as well as Diastolic Pressure. Assume the intercept of the regression model as **5** and the slope of independent variables as **-0.03 (negative)**.

$\alpha = 0.02$

Patient	Systolic Pressure mm Hg	Diastolic Pressure mm Hg	BMI	Waist Circumference Threshold cm	RR-CHD (Relative Risk of Coronary Heart Disease)
1	140	80	35	100	1.81
2	120	80	25	80	1.22
3	130	100	30	60	1.71

$\lambda = 5$

Apply a regularization on the same problem with regularization constant 5 and apply GD for 2 iterations & interpret the results. Try both ridge regression as well as lasso regression. Below equation is changed only for ridge regression. Students must apply appropriate GD update equation changes for this problem.

Hint

Answer

Steps :

1. Identification of the equations y = w0+w1X1+W2X2
2. Cost function & derivative
    1. W0` = w0 – 1/3 * learning rate * (sum (w0+w1X1+W2X2 - y))
    2. W1` = w1* (1- learning rate * regularization constant) – 1/3 * learning rate * (sum (w0+w1X1+W2X2 - y) * x1)
    3. W2` = w2 * (1- learning rate * regularization constant) – 1/3 * learning rate * (sum (w0+w1X1+W2X2 - y)*x2)

$(1 - \alpha\lambda)$

3. Apply the equations

Check

Answer:

Model	w_0	w_1	w_2
Linear	5.0016	0.0477	0.1793
Ridge	5.0016	0.0507	0.1823
Lasso	5.0016	0.1477	0.2793

Interpretation

- **Without Regularization**

Weights increase significantly.

- **Ridge**

Weights shrink slightly due to multiplication by 0.9.

- **Lasso**

Adds fixed penalty → can push weights toward zero faster.

Python Lab Exercise – For Student Practice

Go to your virtual lab file under : Machine Learning ☐ LabCapsule 2 Linear _ Polynomial Regression ☐ 3 Polynomial Regression

Download ML_Lab 5 PolynomialRegression.ipynb

Try to change the degree to {2, 5, 7, 10} in the below function line : PolynomialFeatures(degree=3)

Observe and interpret on the performance of the training data

Note: Training vs Test Data split is not coded in this implementation. Below are the suggestions to experiment further:

- Add few hundreds of data (Use synthetic data generation python libraries available : Refer here)
- Split the data into 80% train vs 20% test set
- Built the polynomial model using above four different degree on training set
- For each of the model apply in the test set
- Find the MSE for both training data and separately for test data
- Plot the four experiment results in a plot and interpret the notion of overfit vs underfit
- Suggest which degree is a best fit .

Thank you